Training Document for Integrated Automation Solutions

Totally Integrated Automation (TIA)

MODULE M1

'Startup' Programming of the

SIMATIC S7-1200 with TIA Portal V10

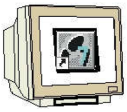
We wish to thank the Michael Dziallas Engineering corporation and the instructors of vocational/professional schools as well as all other persons for their support during the preparation of this document.

PAGE

The following symbols guide you through this module:

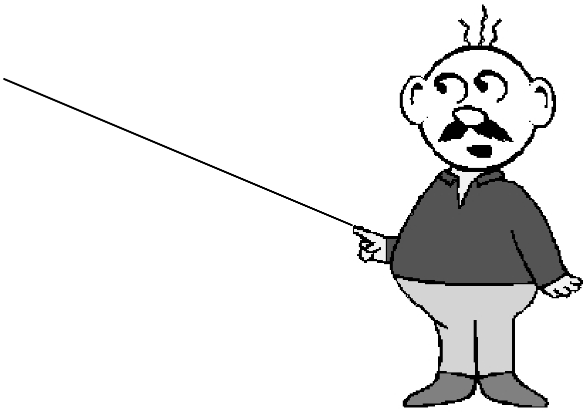

Information


Installation


Programming

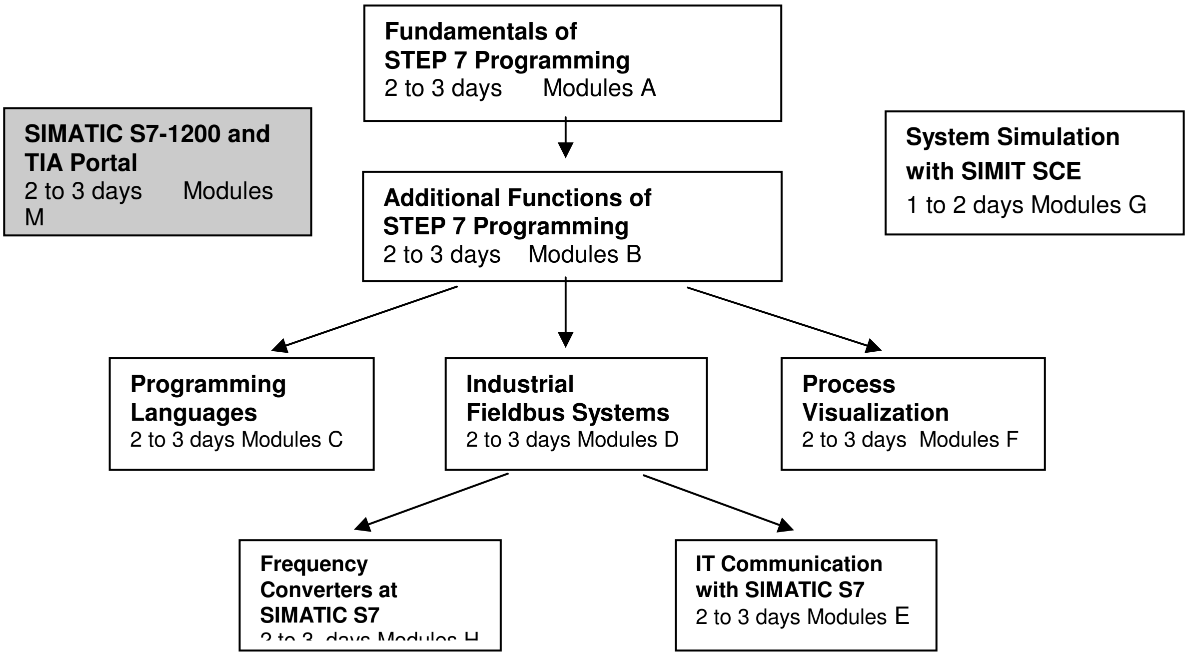

Sample Task


Notes



1. Preface

Regarding its content, module M01 is part of the training unit **'SIMATIC S7-1200 and TIA Portal'** and represents a **fast entry point** for SIMATIC S7 1200 programming.



Training Objective:

In this module M01, the reader learns how to program the programmable logic controller (PLC) SIMATIC S7-1200, using the programming tool TIA Portal. This module provides the fundamentals and demonstrates with the steps listed below how this is done, using a detailed example.

- Installing the software and setting the programming interface
- Explanation: What is a PLC and how does it work
- Structure and operation of the SIMATIC S7-1200 PLC
- Generating, loading and testing a sample program

Preconditions:

To successfully work through this module M01, the following knowledge is assumed:
- How to operate Windows

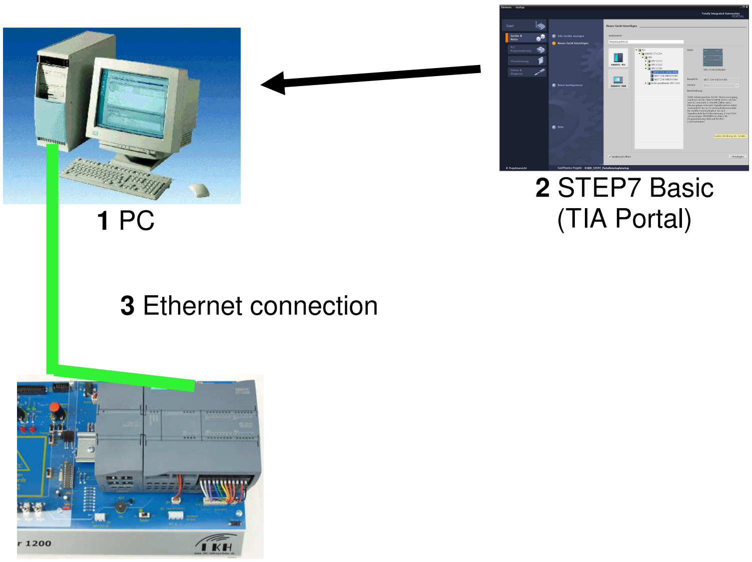
Hardware and software needed

1 PC Pentium 4, 1.7 GHz 1 (XP) – 2 (Vista) GB RAM, free disk storage approx. 2 GB; operating system Windows XP (Home SP3, Professional SP3)/Windows Vista (Home Premium SP1, Business SP1, Ultimate SP1

2 Software STEP7 Basic V10.5 SP1 (Totally Integrated Automation (TIA) Portal V10.5)

3 Ethernet connection between PC and CPU 1214C

4 PLC SIMATIC S7-1200; for example, CPU 1214C. The inputs have to be brought out to a panel.



2. Notes on Programming the SIMATIC S7-1200

2.1 Automation System SIMATIC S7-1200

The automation system SIMATIC S7-1200 is a modular mini-controller system for the lower performance range.
An extensive module spectrum is available for optimum adaptation to the automation task.
The S7 controller consists of a CPU that is equipped with inputs and outputs for digital and analog signals.
Additional input and output modules (IO modules) can be installed if the integrated inputs and outputs are not sufficient for the desired application.
If needed, communication processors for RS232 or RS485 are added.
An integrated TCP/IP interface is obligatory for all CPUs.

With the S7 program, the programmable logic controller (PLC) monitors and controls a machine or a process, whereby the IO modules are polled in the S7 program by means of the input addresses (%I) and addressed by means of output addresses (%Q).

The system is programmed with the software STEP 7 Basic V10.5.

2.2 Programming Software STEP 7 Basic V10.5 (TIA Portal V10.5)

The software STEP 7 Basic V10.5 is the programming tool for the automation system

- SIMATIC S7-1200

With STEP 7 Basic V10.5, the following functions can be utilized to automate a plant:

- Configuring and parameterizing the hardware
- Defining the communication
- Programming
- Testing, commissioning and service with the operating/diagnostic functions
- Documentation
- Generating the visual displays for the SIMATIC basic panels

All functions are supported with detailed online help.

3. Installing the Software STEP 7 Basic V10.5 (TIA Portal V10.5)

STEP 7 Basic V10.5 is supplied on a DVD.

To install STEP 7 Basic V10.5, do the following:

1. Insert the DVD of STEP 7 Basic V10.5 in the DVD drive.
2. The setup program is started automatically. If not, start it by double clicking on the file '→ **START.exe**'.
   The setup program guides you through the entire installation of STEP 7 Basic V10.5.

   To utilize STEP 7 Basic V10.5, no license key or dongle is needed on your computer.

4. Connecting to the CPU by means of TCP/IP, and Resetting to Factory Setting

To program the SIMATIC S7-1200 from the PC, the PG or a laptop, you need a TCP/IP connection.

For the PC and the SIMATIC S7-1200 to communicate with each other, it is important also that the IP addresses of both devices match.

First, we show you how to set the computer's IP address.

1. From the '**System control'**, call the **'Network connections'**. Then, select the **'Properties'** of the LAN connection (→ Start → Settings → System control → Network connections→ Local Area Connection → Properties)

2. Select the **'Properties**' from the **'Internet Protocol (TCP/IP)'** (→ Internet Protocol (TCP/IP) → Properties)

3. You can now set the **'IP address'** and the **'Subnet screen form'**, and accept with **'OK'** (→ Use the following IP address → IP address: 192.168.0.99 → Subnet screen form 255.255.255.0 → OK → Close)

Notes on networking on the Ethernet (additional information is provided in Appendix V of the training document):

MAC address:
The MAC address consists of a permanent and a variable part. The permanent part ("Basic MAC Address") identifies the manufacturer (Siemens, 3COM, ...). The variable part of the MAC address differentiates the various Ethernet stations and should be assigned uniquely world-wide. On each module, a MAC address is imprinted specified by the factory.

Value range for the IP-address:
The IP address consists of 4 decimal numbers from the value range 0 to 255, separated by a period. For example, 141.80.0.16

Value range for the subnet screen form:
This screen form is used to recognize whether a station or its IP address belongs to the local subnetwork, or can be accessed only by means of a router.
The subnet screen form consists of four decimal numbers from the value range 0 to 255, separated by a period. For example, 255.255.0.0
In their binary representation, the 4 decimal numbers of the subnet screen form have to contain -from the left- a series of gapless values "1" and from the right a series of gapless values "0".
The values "1" specify the area of the IP address for the network number. The values "0" specify the area of the IP address for the station address.
Example:

Correct values:	255.255.0.0 Decimal = 1111 1111.1111 1111.0000 0000.0000 0000 binary
	255.255.128.0 Decimal = 1111 1111.1111 1111.1000 0000.0000 0000 binary
	255.254.0.0 Decimal = 1111 1111.1111 1110.0000 0000.0000.0000 binary
Wrong value:	255.255.1.0 Decimal = 1111 1111.1111 1111.0000 000***1***.0000 0000 binary

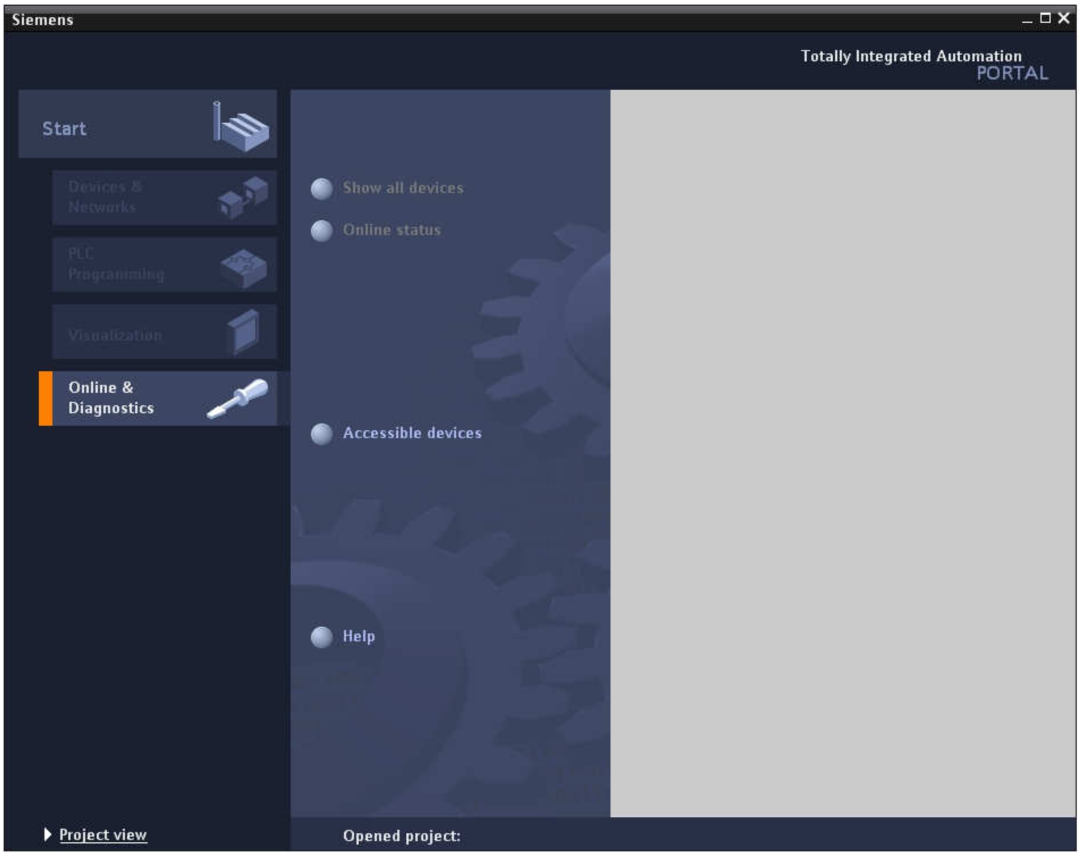
Value range for the address of the gateway (Router):
The address consists of 4 decimal numbers from the value range 0 to 255, separated by a period. For example, 141.80.0.1.

Relationship of IP addresses, router address, and subnet screen form:
The IP address and the gateway address are to differ only at positions where an "0" is located in the subnet screen form.
Example:
You entered the following: for the subnet screen form 255.255.255.0, for the IP address 141.30.0.5 and for the router address 141.30.128.1.
The IP address and the gateway address must have a different value only in the 4th decimal number. However, in the example, the 3rd position already differs.
That means, in the example you have to change alternatively:
- the subnet screen form to: 255.255.0.0 or
- the IP address to: 141.30.128.5 or
- the gateway address to: 141.30.0.1

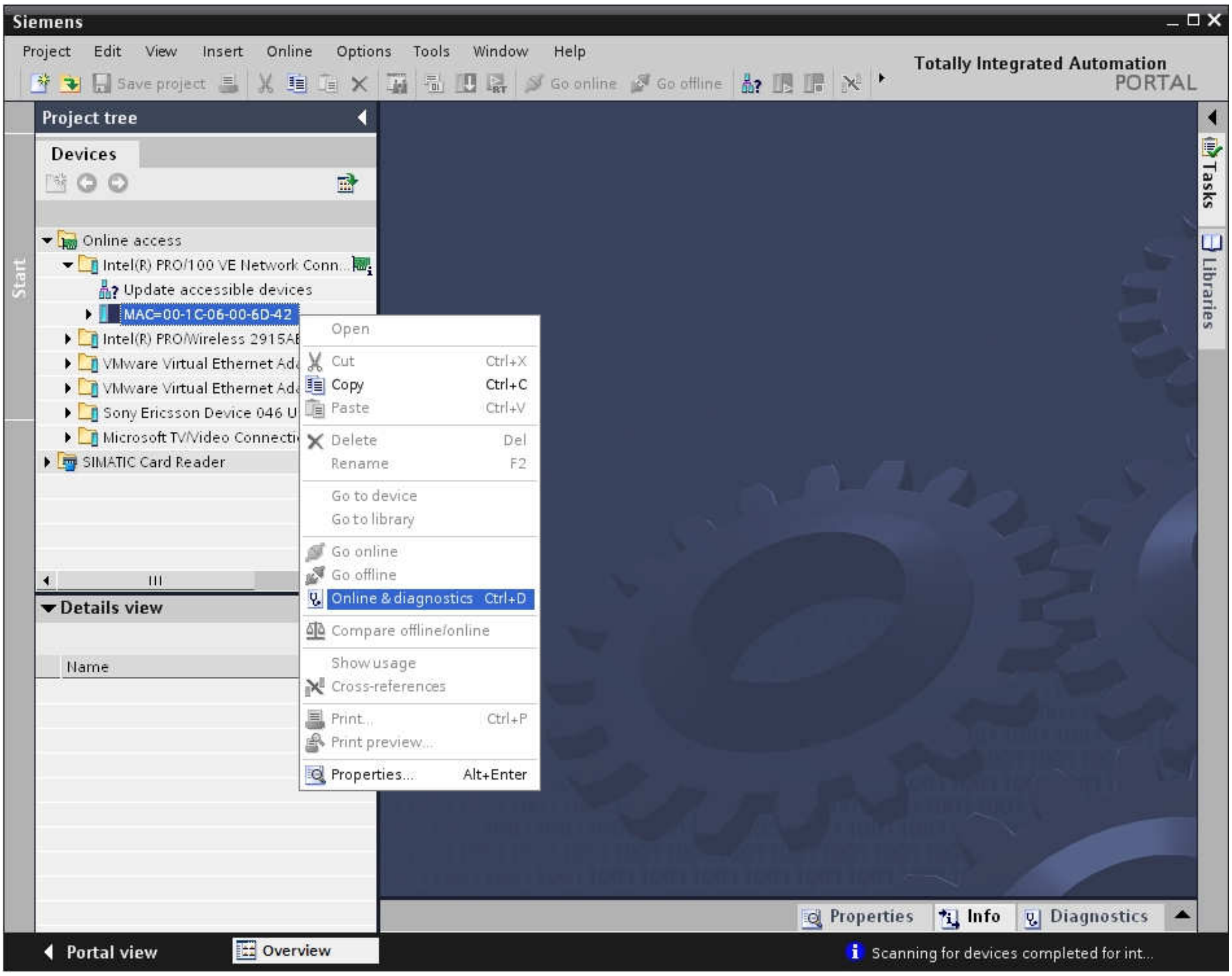
The SIMATIC S7-1200 IP address is set as follows:

4. Select the **'Totally Integrated Automation Portal'**; it is called here with a double click (→ Totally Integrated Automation Portal V10)

5. Then, select the **'Project View'**. (→ Project view)

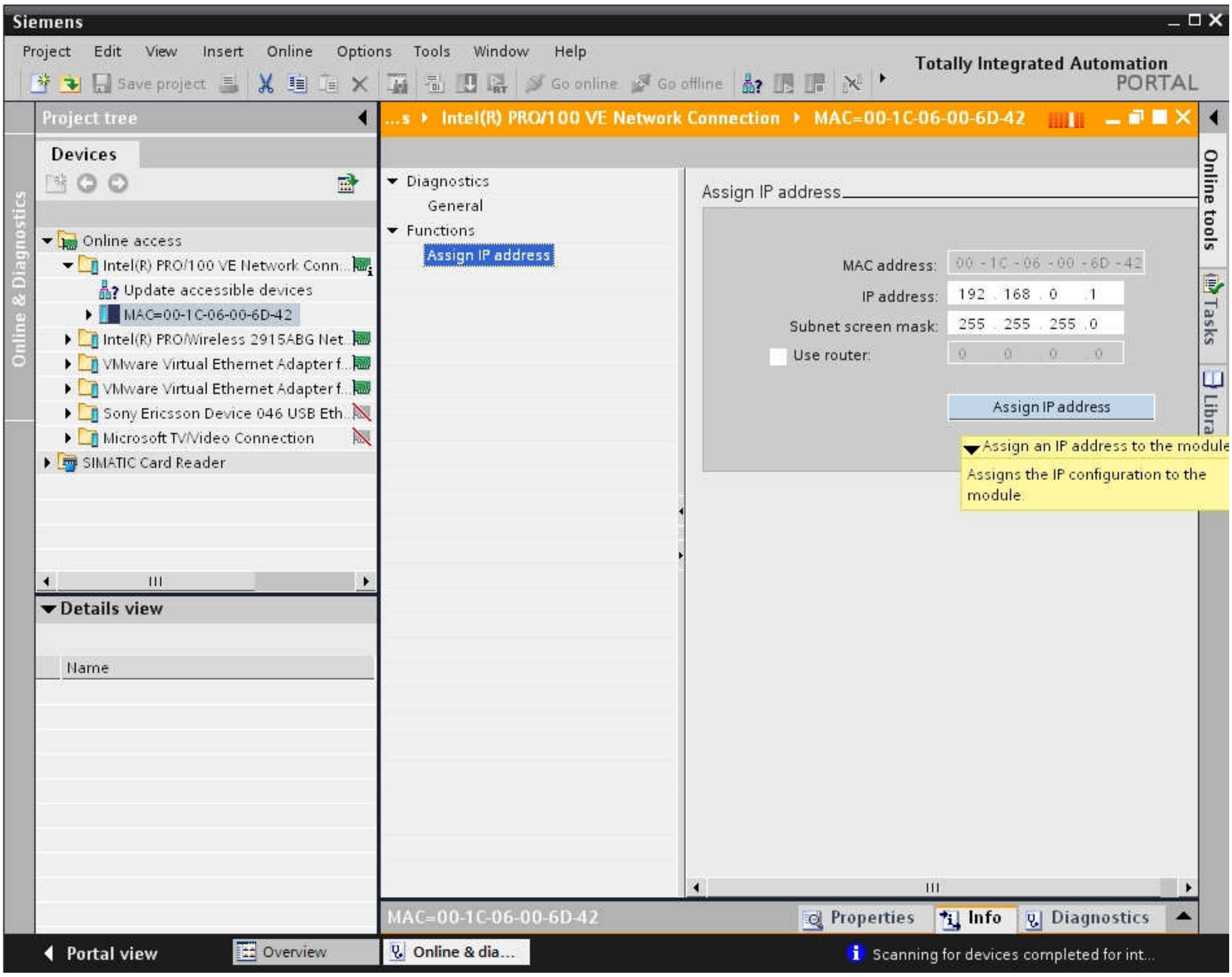


6. Next, in project navigation, select under **'Online accesses'** the network card that was already set beforehand. If you click here on **'Update accessible stations'** <<Erreichbare Teilnehmer aktualisieren>>, you will see the the MAC address of the connected SIMATIC S7-1200. Select **'Online & Diagnosis'**. (→ Online accesses → … Network Connection → Update accessible stations → MAC= ….. → Online & Diagnosis)

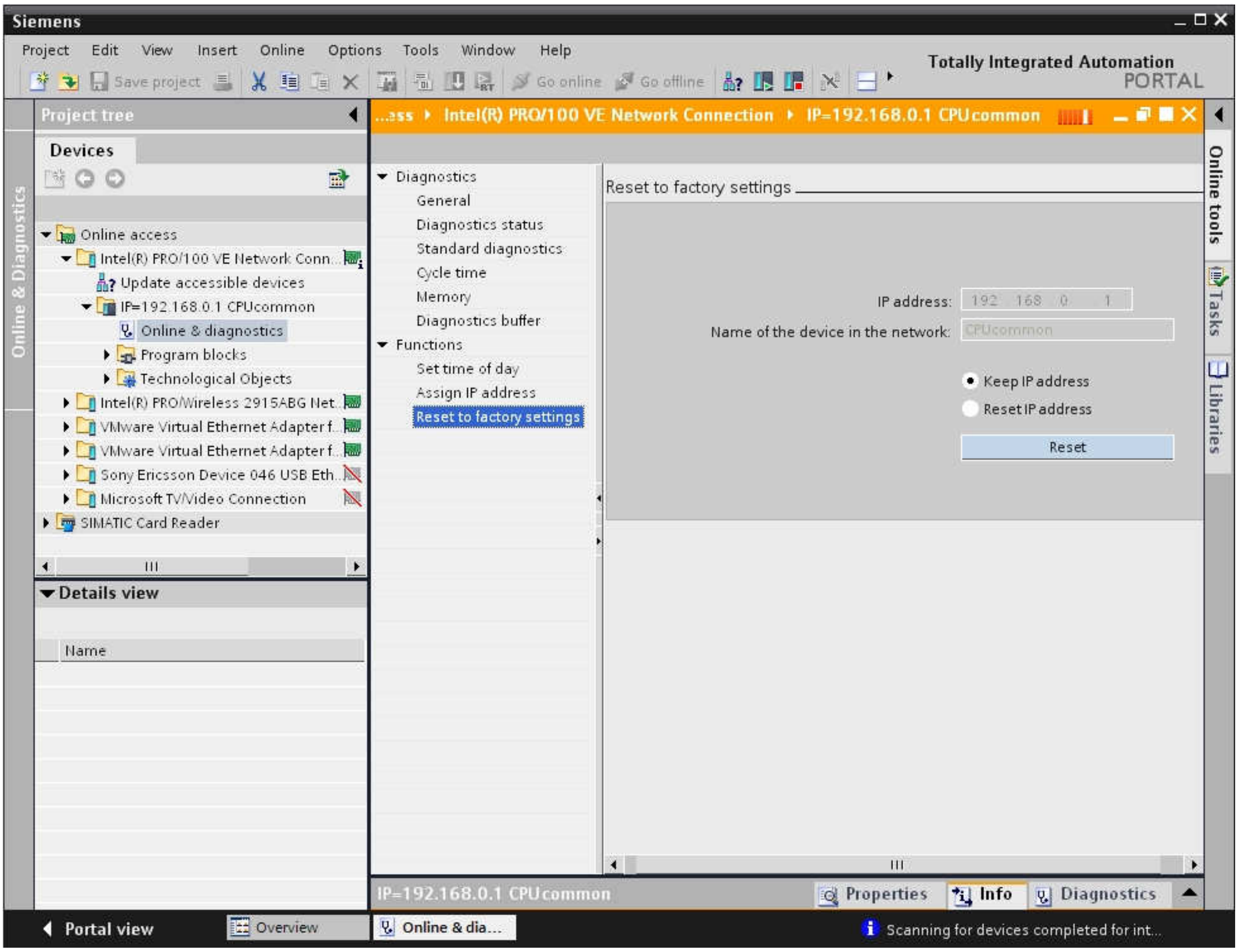


Note: If an IP address was set previously at the CPU, you will see this address instead of the MAC address.

7. Under **’Functions’** you will see the item **’Assign IP address’**. Here, enter **’IP address’** and **’Subnet screen form’**. Then, click on **’Assign IP address’**, and your SIMATIC S7-1200 will be assigned this new address (→ Functions → Assign IP address → IP address: 192.168.0.1 → Subnet screen form: 255.255.255.0 → Assign IP address)



8. Under **'Functions',** select **'Reset to factory settings'**. Keep this setting on **'Retain IP address'** and click on **'Reset'**. (→ Functions → Reset to factory settings → Retain IP address → Reset)



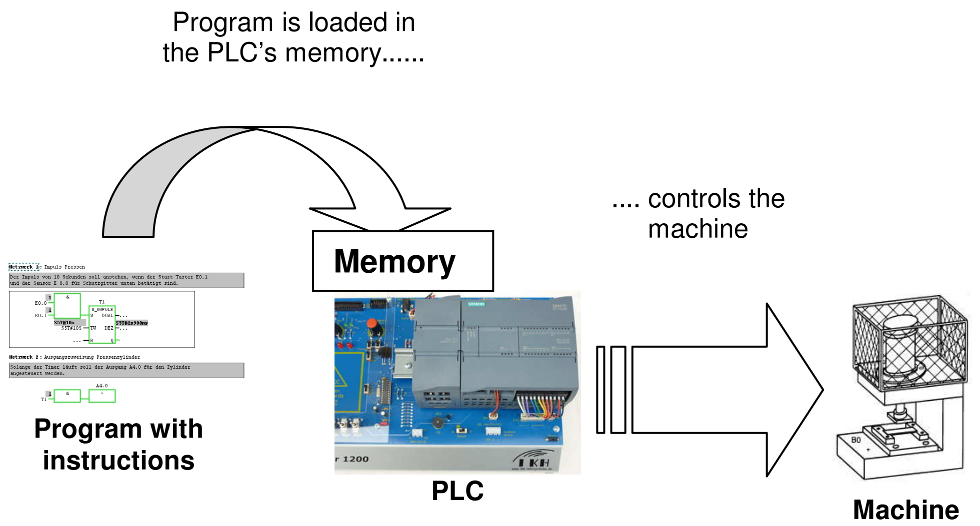
9. Confirm the query whether you want to go through with a reset to the factory setting with **'OK'** (→ OK)

5. What is a PLC and What are PLCs Used For?

5.1 What does the term PLC mean?

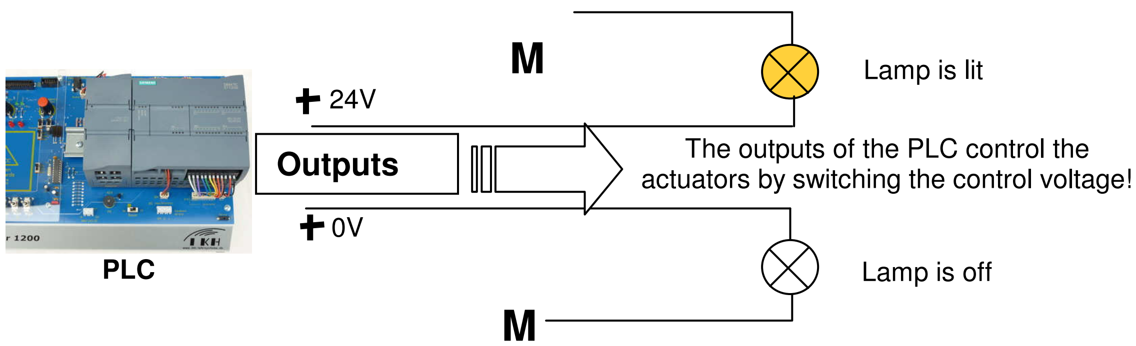
PLC is the abbreviation for **P**rogrammable **L**ogic **C**ontroller. It describes a device that controls a process (for example, a printing press for printing newspapers, a filling plant for filling cement in bags, a press for forming plastic shapes, etc....). This is performed according to the instructions of a program that is located in the memory of the device.

.



5.2 How does the PLC Control the Process?

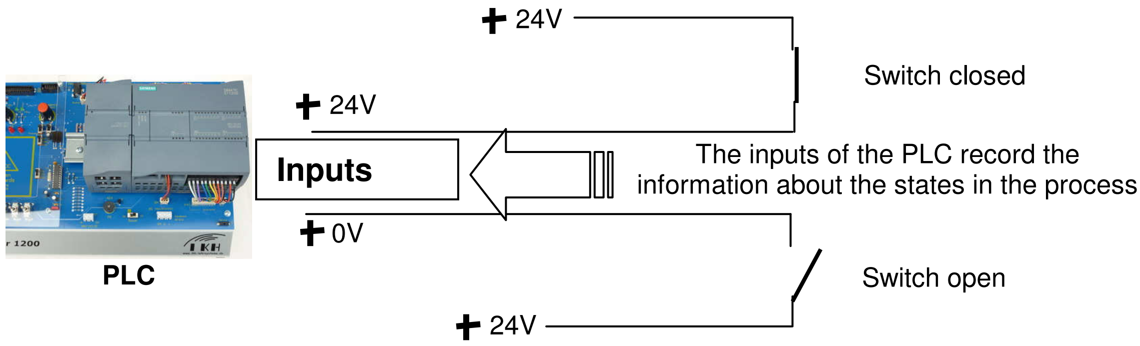
The PLC controls the process as follows: through the PLC connections called **outputs**, so-called **actuators** are wired with a control voltage of 24V for example. This allows for switching motors on and off, opening and closing valves, turning lamps on and off.



5.3 How does the PLC Get the Information about the Process States?

The PLC receives information about the process from the so-called **signal transmitters** that are wired to the **inputs** of the PLC. These signal transmitters can be, for example, sensors that recognize whether a work piece is in a certain position, or they can be simple switches and pushbuttons that may be open or closed. Here, we differentiate between **break contact elements** that are closed if not operated, and **make contact elements** that are open if not activated.

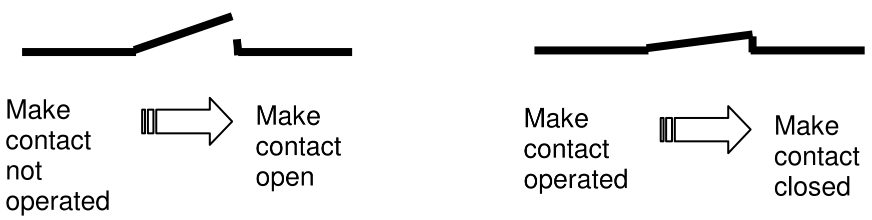
,



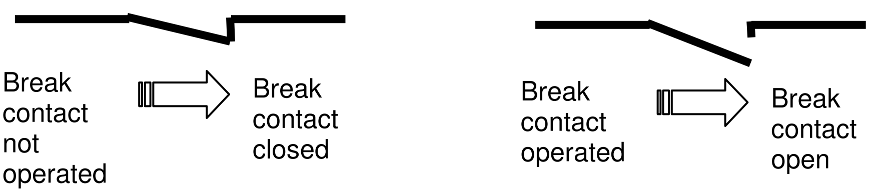
5.4 What is the Difference between Break Contact Elements and Make Contact Elements?

As to the signal transmitters, we differentiate between **break contacts** and **make contacts**.

The switch shown below is a make contact; i.e., it is closed exactly when it was operated

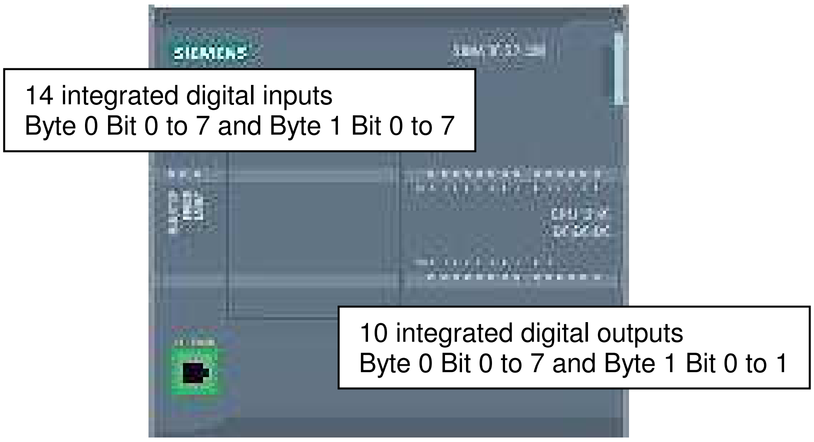


The switch shown below is a break contact; i.e, it is closed exactly when it was not operated.



5.5 How does the SIMATIC S7-1200 Address Individual Input/Output Signals?

Specifying a certain input or output within the program is called addressing.
The PLC inputs and outputs are usually combined into groups of 8 on digital input modules and digital output modules. This unit of 8 is called a **byte**. Each such group receives a number as the so-called **byte address**.
In order to address a single input or output within a byte, each byte is broken down into **bits**. These are numbered Bit 0 to Bit 7. This is how we arrive at the **bit address**.
The PLC shown here has the input bytes 0 and 1 as well as the output bytes 4 and 5.



To address the fifth digital input, for example, we specify the following address:

%E 0 . 4

%E here indicates the address type as input, **0** the byte address, and **4** the bit address.
Byte address and bit address are always separated by a period.

Note: For the bit address, a **4** is shown for the fifth input, because we start counting with 0.

To address the 10th output, for example, we specify the following address:

%A 1 . 1

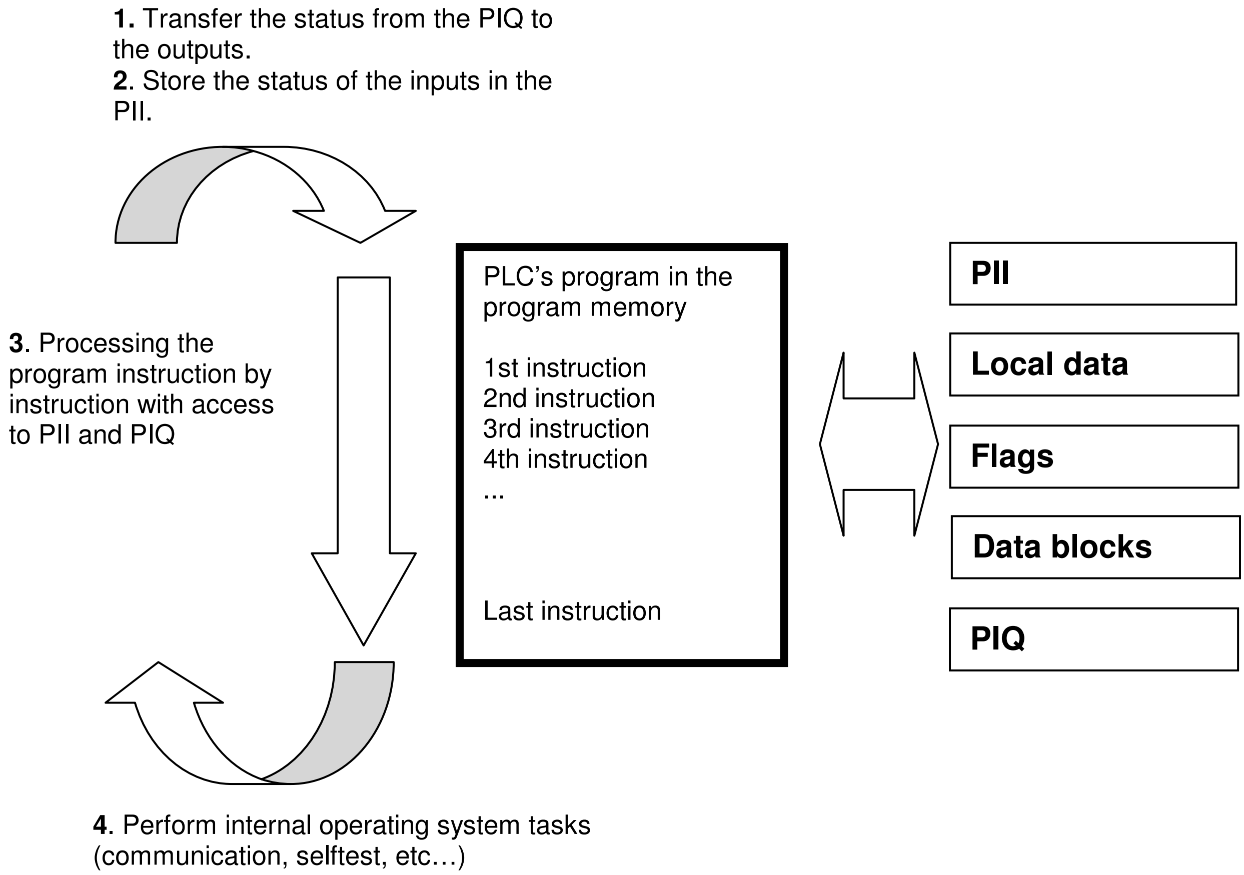
%A here indicates the address type as output, **1** the byte address, and **1** the bit address.
Byte address and bit address are always separated by a period.

Note: For the bit address, a **1** is shown at the 10th output, because we start counting with 0..

5.6 How is the Program Processed in the PLC?

The program is processed in the PLC cyclically, in the following sequence:

1. First, the status is transferred from the process image of the outputs (**PIQ**) to the outputs, and switched on or off.
2. Then the **processor** -which is practically the PLC's brain- inquires whether the individual inputs are carrying voltage. This status of the inputs is stored in the process image of the inputs (**PII**). For the inputs that carry voltage, the information 1 or "High“ is stored, for those that don't the information 0 or "Low“.
3. This processor then processes the program stored in the program memory. The program consists of a list of logic operations and instructions that are processed one after the other. For the required input information, the processor accesses the PII that was entered previously, and the result of the logic operation (RLO) is written into a process image of the outputs (**PIQ**). If necessary, the processor also accesses other memory areas during program processing; for example, for local data of sub-programs, data blocks and flags.
4. Then, internal operating system tasks such as self tests and communication are performed Then we continue with Item 1.



Note: The time the processor needs for this sequence is called cycle time. In turn, the cycle time depends on the number and type of instructions and the processor capacity.

5.7 What do Logic Operations Look Like in the PLC Program?

Logic operations are used to specify conditions for switching an output.
In the PLC program, these can be programmed in the programming languages Ladder Diagram (**LAD**) or Function Block Diagram (**FBD**).
We will use **FBD** for illustration.
There is a large number of logic operations that can be used in PLC programs.
However, **AND** as well as **OR** operations and the **NEGATION** of an input are used most frequently and are explained briefly below, using examples.

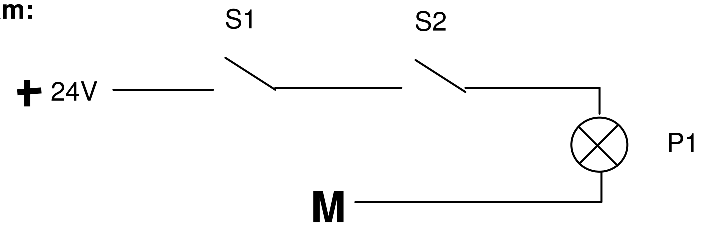
Note: Clearly arranged Information about additional logic operations can be obtained quickly in online help.

5.7.1 AND Operation

Example of an AND operation:

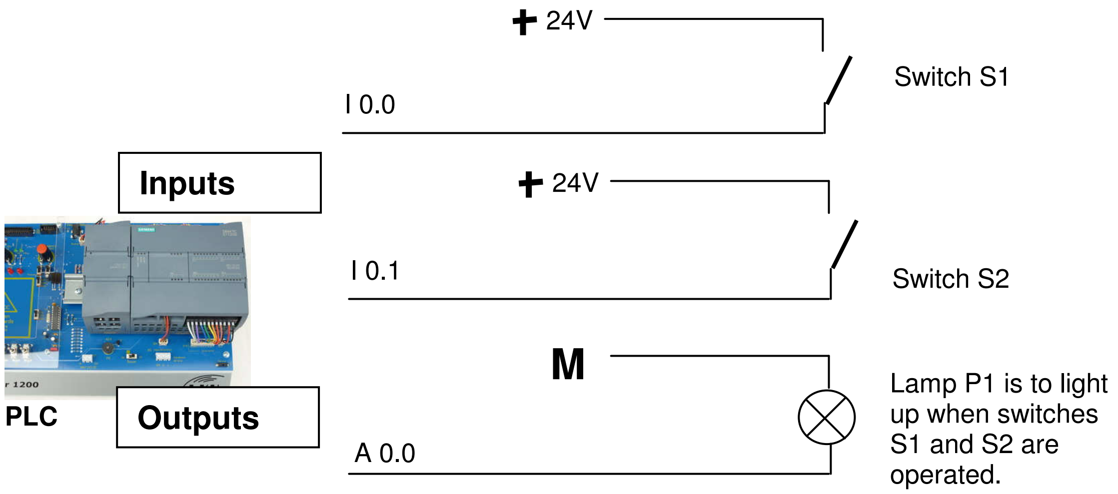
A lamp is to light up when two switches are operated simultaneously as make contacts.

Diagram:



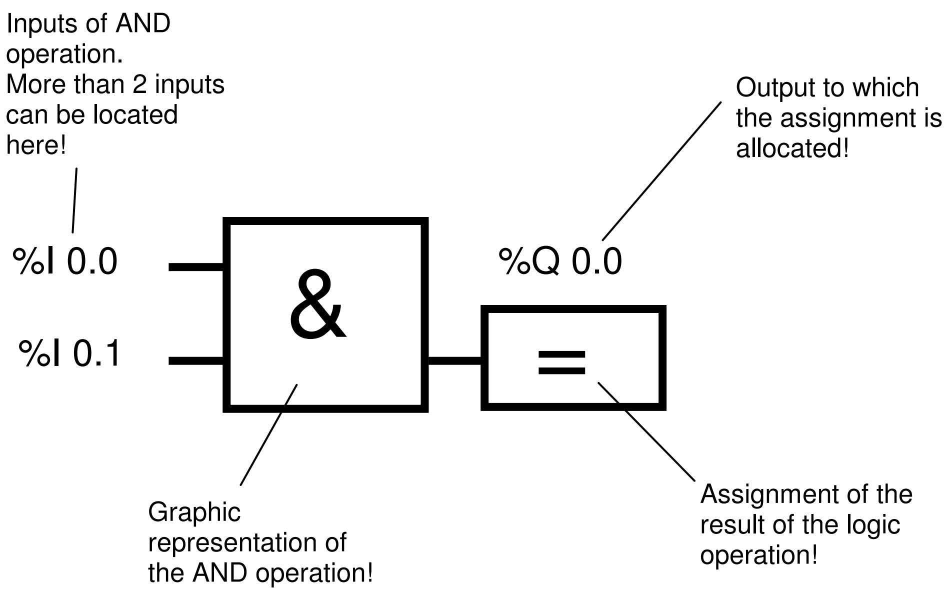
Explanation:
The lamp lights up exactly when both switches are operated.
That is, when switches S1 **and** S2 are operated, lamp P1 is lit.

Wiring the PLC:
To apply this logic to a PLC program, both switches have to be connected to inputs of the PLC. Here, S1 is wired to input I 0.0 and S2 to input I 0.1.
In addition, lamp P1 has to be connected to an output; for example Q 0.0.



AND logic in the FBD:

In the function block diagram FBD, the AND operation is programmed using a graphic representation, and looks like this:

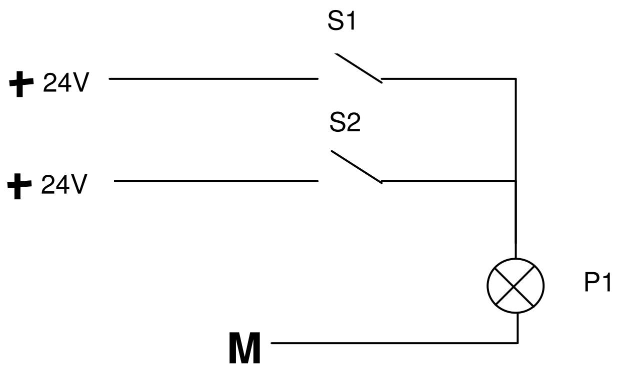


5.7.2 OR Operation

Example of an OR operation:

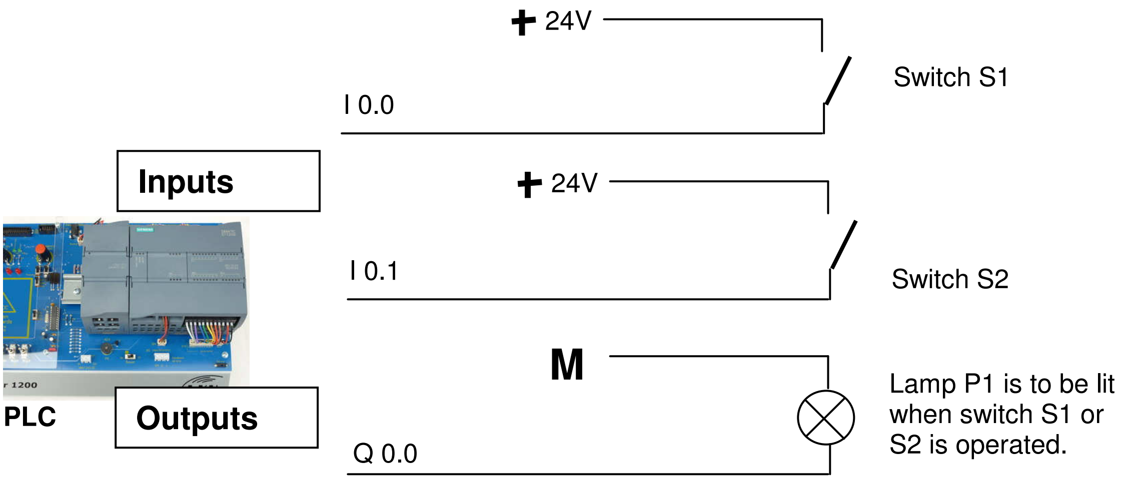
A lamp is to light up when one or both <<?>> of two switches are operated as make contacts.

Diagram:



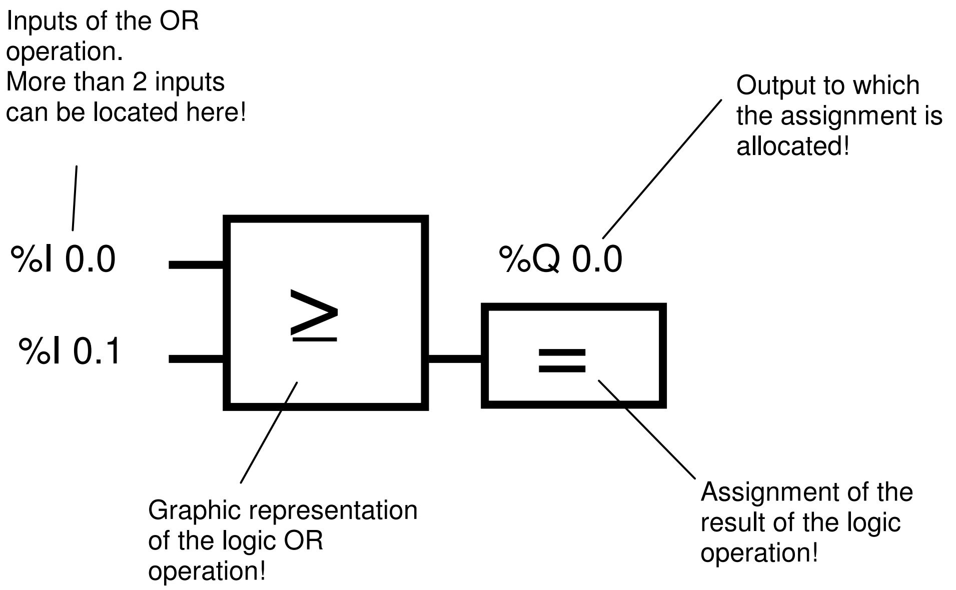
Explanation:
The lamp lights up exactly when one or both switches are operated.
That is, if switch S1 **or** S2 is operated, lamp P1 is lit.

Wiring the PLC:
To apply this logic to a PLC program, both switches have to be connected to inputs at the PLC, of course. Here, S1 is connected to input I 0.0 and S2 to input I 0.1.
In addition, lamp P1 has to be connected to an output; for example, Q 0.0.



OR operation in the FBD:

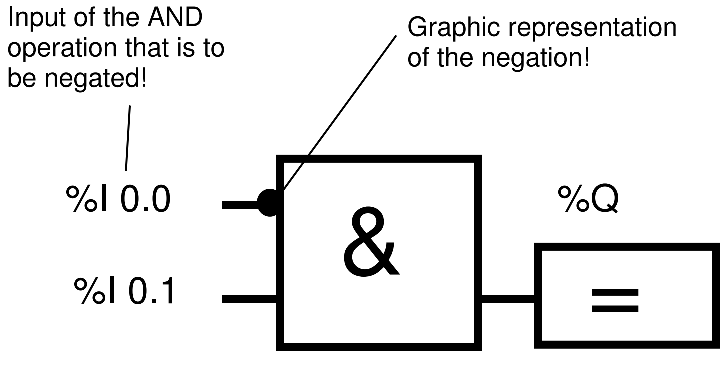
In the function plan FBD, the OR operation is programmed using a graphic representation, and looks like this:



5.7.3 Negation

In logic operations, it is often necessary to inquire whether a **make contact** was **NOT operated** or whether a **break contact** was **operated** and thus no voltage is applied to the corresponding input. This happens when we program a **Negation** at the input of the AND or OR operation.

In the function block diagram FBD, the negation of an input at an AND operation is programmed with the following graphic representation:

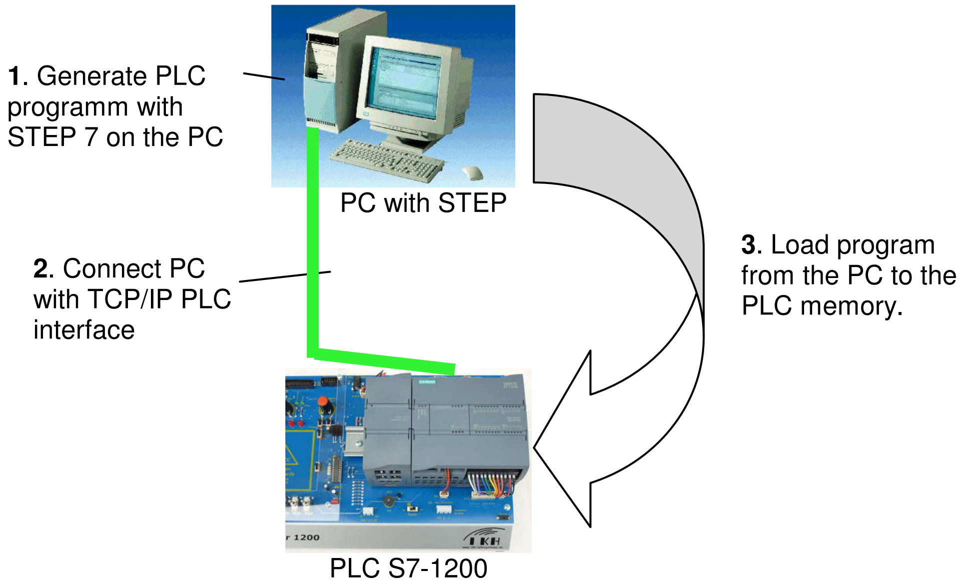


That means, voltage is applied to the output %Q 0.0 exactly when %I 0.0 is not connected and %I 0.1 is connected.

5.8 How is the PLC Program generated? How does it get to the PLC’s Memory?

The PLC program is generated on a PC using the software STEP 7, and temporarily stored there. After the PC is connected with the TCP/IP interface of the PLC, the program can be transferred with a load function to the PLC’s memory.

The PC is no longer needed for further program processing in the PLC.



Note: The exact sequence is described step by step in the chapters below.

6. Configuring and Operating the SIMATIC S7-1200

Module Spectrum:

The SIMATIC S7-1200 is a modular automation system and offers the following module spectrum:

- Central modules CPU with different capacity, integrated inputs/outputs and PROFINET interface (for example, CPU1214C)

- Power supply PM with input AC 120/230V, 50Hz/60Hz, 1.2A/0.7A, and output DC 24V/2.5A

- Signal boards SB for adding analog or digital inputs/outputs; whereby the size of the CPU does not change
  .
  (signal boards can be used with the CPUs 1211C/1212C and 1214C)

- Signal modules SM for digital and analog inputs and outputs
  (for CPUs 1212C a maximum of 2 SMs can be used, for 1214C a maximum of. 8)

- Communication modules CM for serial communication RS 232/RS 485
  (for CPUs 1211C/1212C and 1214C, up to 3 CMs can be used)

- Compact Switch Module CSM with 4x RJ45 socket connectors 10/100 MBit/s

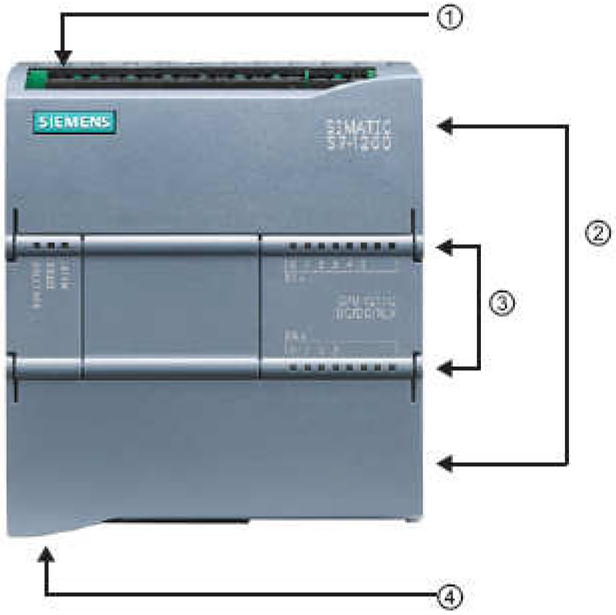
SIMATIC memory cards 2MB or 24MB for storing program data and simple CPU replacement for maintenance

Note: For this module M01, any CPU with integrated digital inputs and digital outputs is sufficient.

Important CPU elements:

With an integrated voltage supply (24V connection) and integrated inputs and outputs, the S7-1200 CPU is ready, without additional components.

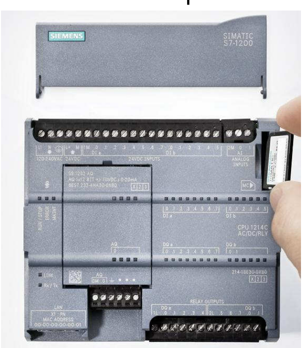
To communicate with a programming device, the CPU is equipped with an integrated TCP/IP port. By means of an ETHERNET network, the CPU is able to communicate with HMI operator devices or other CPUs.



□ Connection 24V
□ Insertable terminal block for user wiring (behind the cover plates)
□ Status LEDs for the integrated I/O and the CPU's operating mode
□ TCP/IP connection (on the lower side of the CPU)

The **SIMATIC Memory Card (MC)** stores the program, data, system data, files and projects. It can be used for the following:

- Transferring a program to several CPUs
- Firmware update of CPUs, signal modules SM and communication modules CM



Operating Modes of the CPU

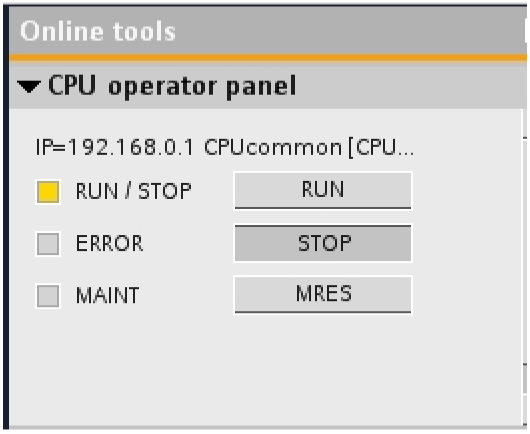
The CPU has the following operating modes:

● In the operating mode **STOP**, the CPU does not execute the program, and you can load a project

● In the operating mode **STARTUP**, the CPU performs a startup.

● In the operating mode **RUN**, the program is executed cyclically. Projects can not be loaded in the CPU's RUN mode.

The CPU does not have a physical switch for changing the operating mode. The operating mode (**STOP** or **RUN**) is changed by using the button on the operator panel of the software STEP7 Basic. In addition, the operator panel is provided with the button **MRES** to perform a general memory reset and displays the status LEDs of the CPU.



The color of the **status LED RUN/STOP** on the front of the CPU indicates its current operating mode.

● **Yellow** light indicates the **STOP** mode.

● **Green** light indicates the **RUN** mode.

● **Blinking** light indicates the **STARTUP** mode.

In addition, there are the LEDs **ERROR** to indicate errors and **MAINT** to indicate that maintenance is required.

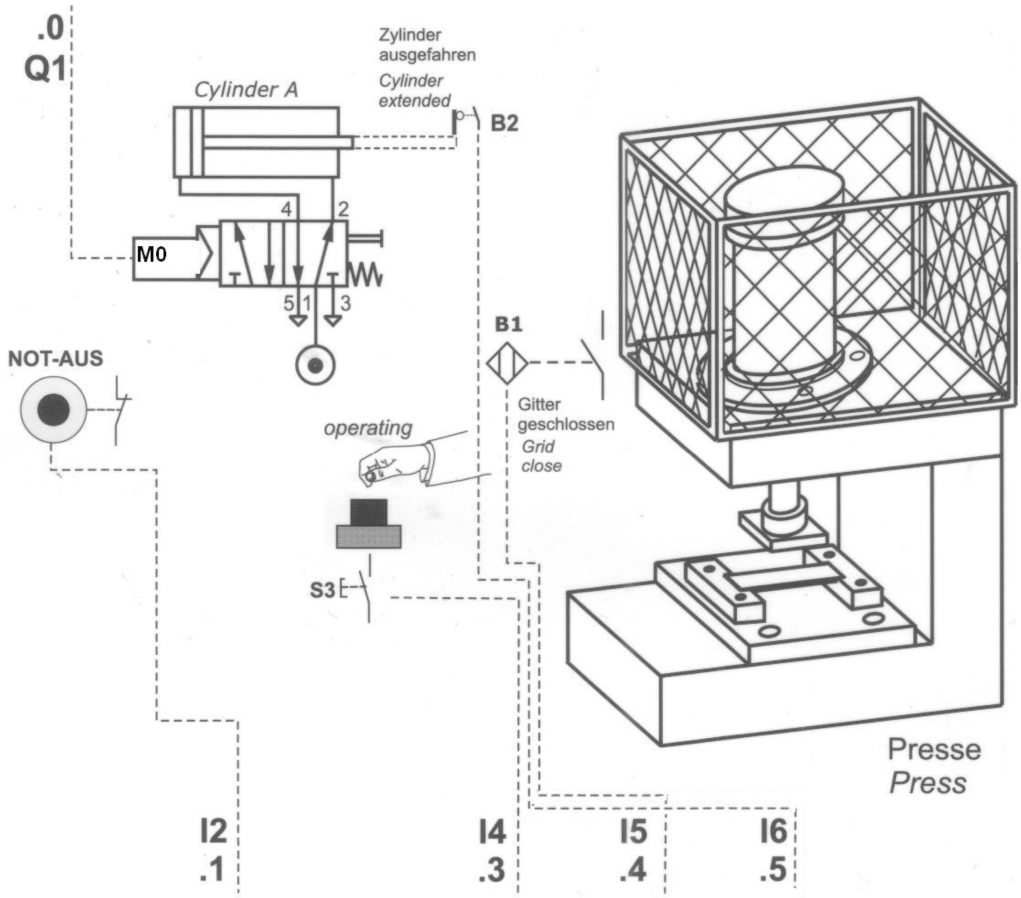
7. Sample Task: Controlling a Press

Our first program consists of programming a press control.

A press with a protective guard is to be activated with a START button S3 only if the protective screen is closed. This state is monitored with a sensor Protective Screen Closed B1.
If this is the case, a 5/2 way valve M0 for the press cylinder is set so that a plastic shape can be pressed.
The press is to retract again when the EMERGENCY OFF button (break contact) EMERGENCY OFF is activated, or the sensor Protective Screen B1 no longer responds, or the sensor Cylinder B2 Extended responds.

Assignment list:

Address	Symbol	Comment
%I 0.1	EMERGENCY OFF	EMERGENCY OFF button (break contact)
%I 0.3	S3	Start button S3 (make contact)
%I 0.4	B1	Sensor protective screen closed (make contact)
%I 0.5	B2	Sensor Cylinder extended (make contact)
%Q 0.0	M0	Extend Cylinder A



8. Programming the Press for the SIMATIC S7-1200

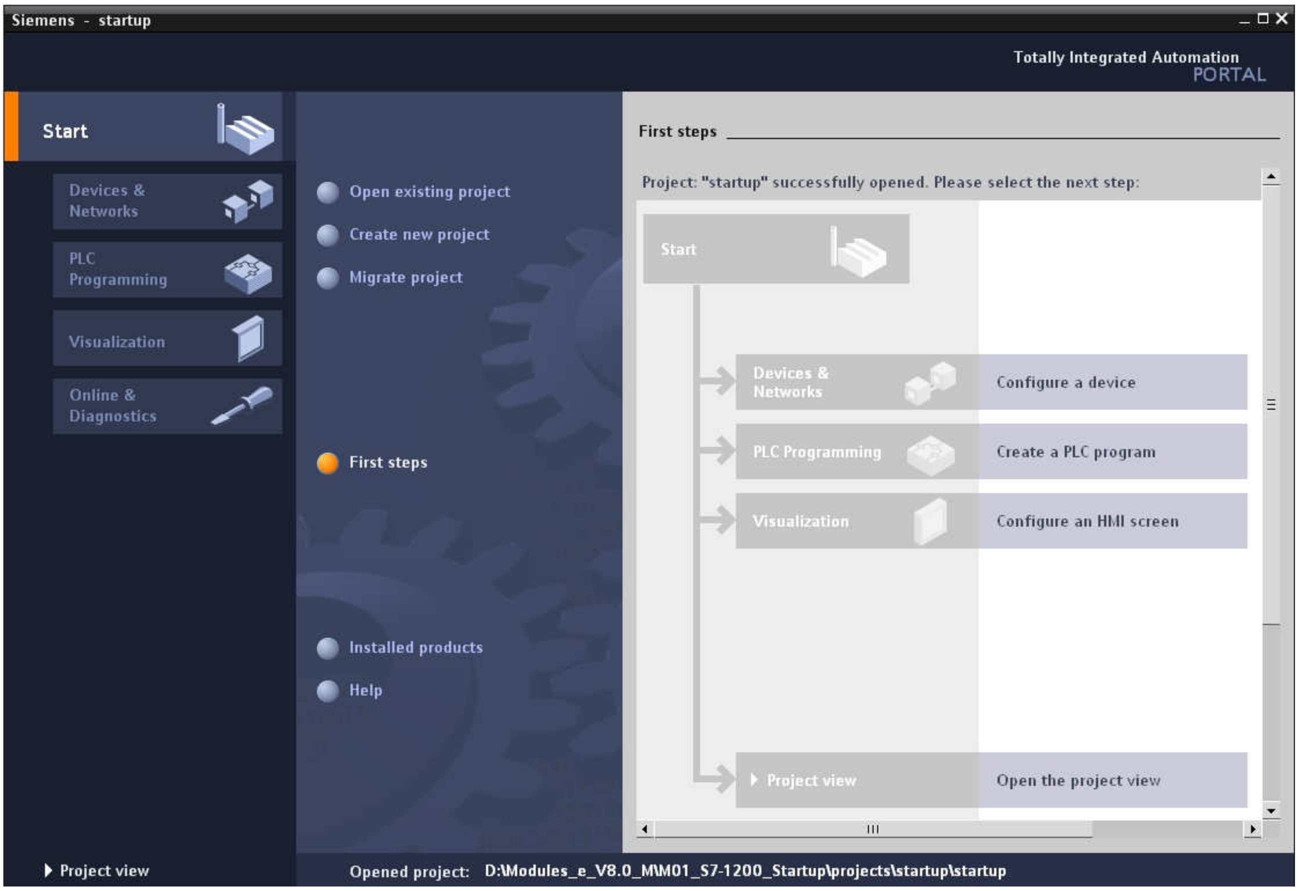
The software **'Totally Integrated Automation Portal'** manages the project and does the programming.

Here, under a uniform interface, the components such as the controller, visualization and networking the automation solution are set up, parameterized and programmed.
Online tools are provided for error diagnosis.

The software **'Totally Integrated Automation Portal'** has two different views: the portal view and the project view.

8.1. Portal View

The portal view provides a task oriented view of the tools for processing the project. Here, you can quickly decide what you want to do, and call the tool for the respective task. If necessary, a change to the project view takes place automatically for the selected task. Primarily, getting started and the first steps are to be facilitated here.

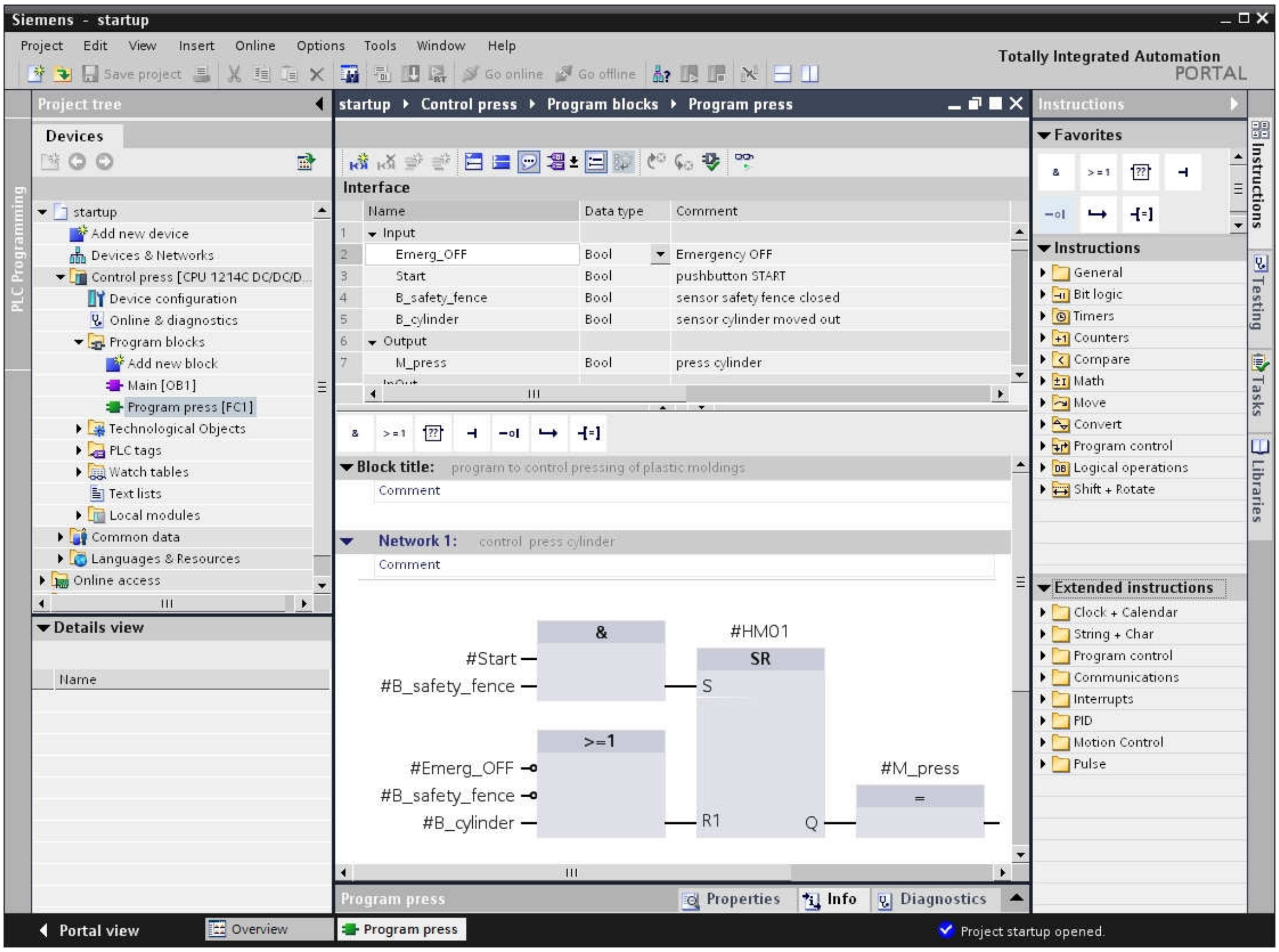


Note: On the lower left, you can jump from the portal view to the project view!

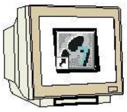
8.2. Project View

The project view is a structured view to all constituent parts of the project. As a matter of standard, the menu bar with the function bars is located on top, project navigation with all the parts of a project on the left, and the task cards (with instructions and libraries, for example) on the right.

If an element (here, for example, program block FC1) is selected in project navigation, it is displayed in the center and can be processed there.



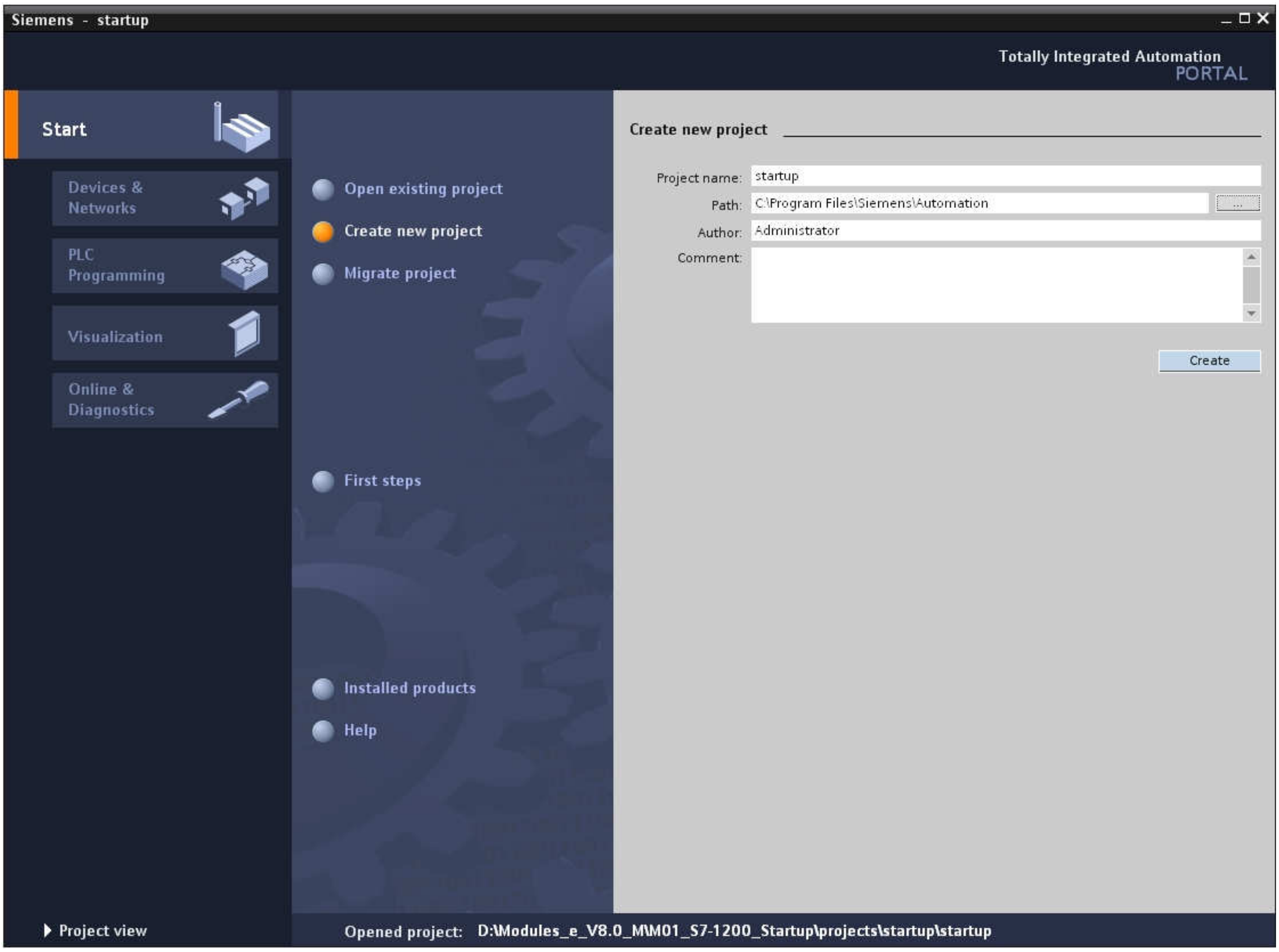
Note: On the lower left, you can jump from the project view to the portal view!

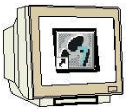


With the following steps, we are setting up a project for the SIMATIC S7-1200 and we are programming the solution of the task:

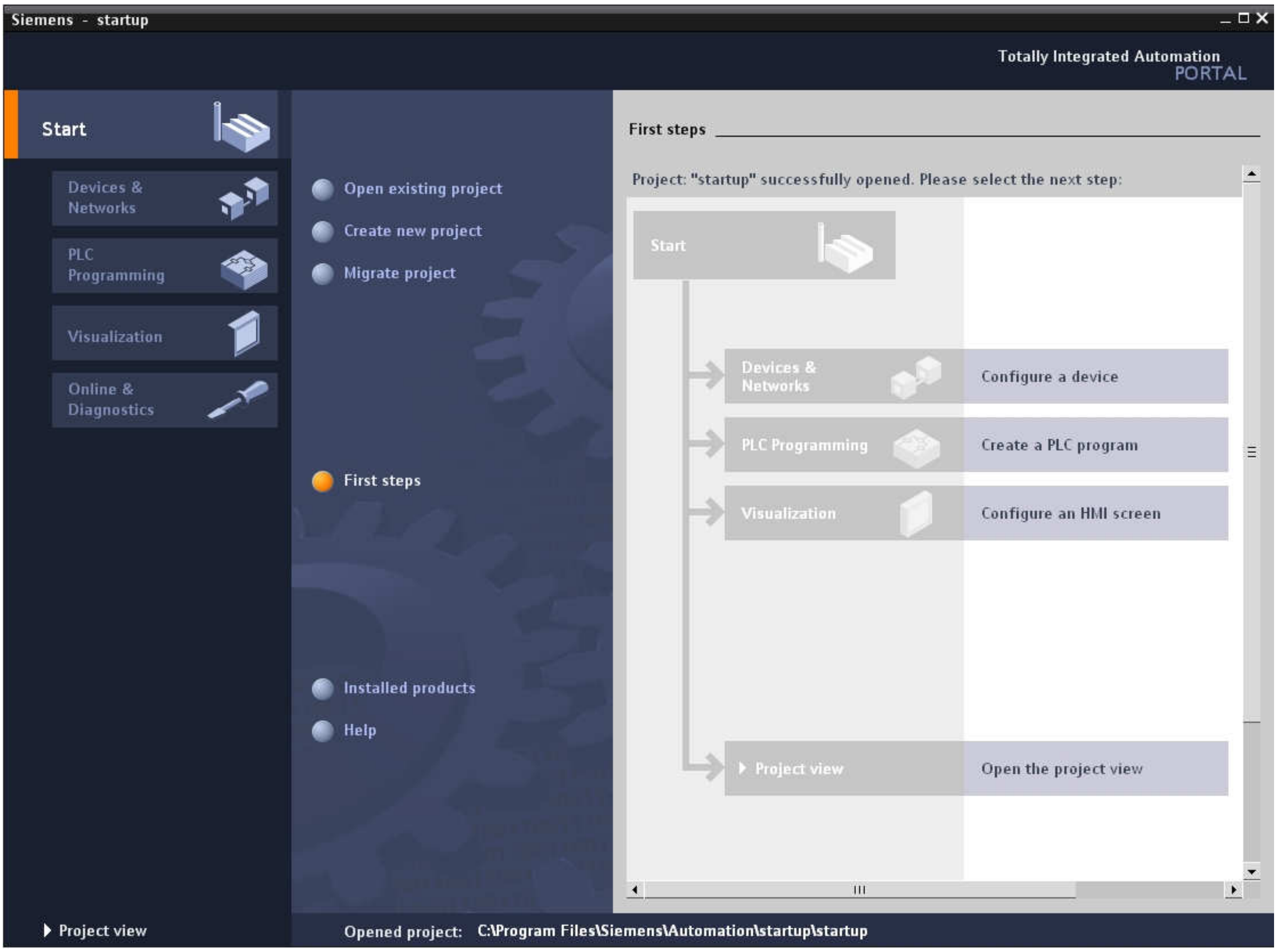
1. The central tool is the **'Totally Integrated Automation Portal'** which is called here with a double click (→ Totally Integrated Automation Portal V10)

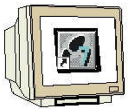
2. Programs for the SIMATIC S7-1200 are managed in projects. Such a project is now set up in the portal view (→ Generate new project → startup → Generate)



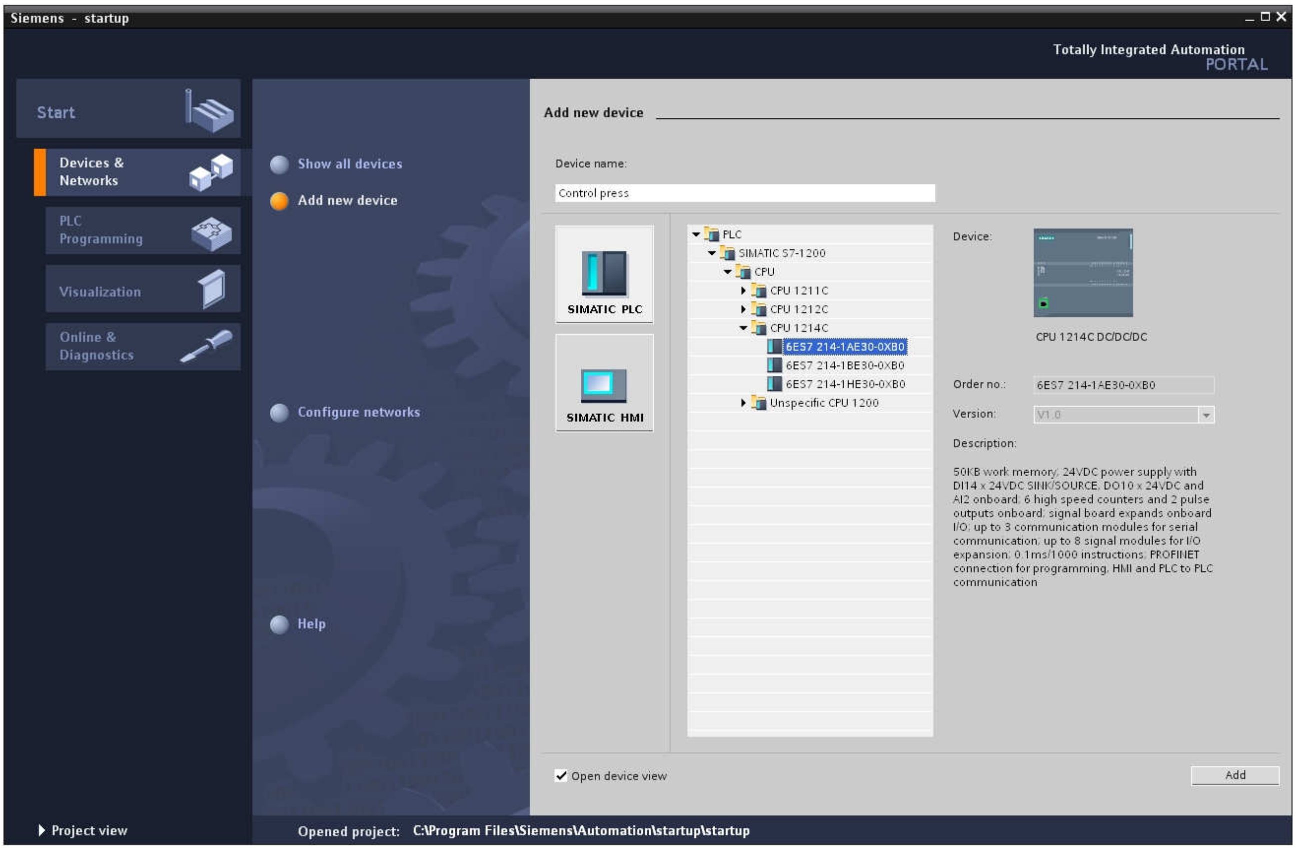


3. Now **'First Steps'** are suggested for the configuration. First, we want to **'Configure a device'** (→ First steps → Configuring a device)

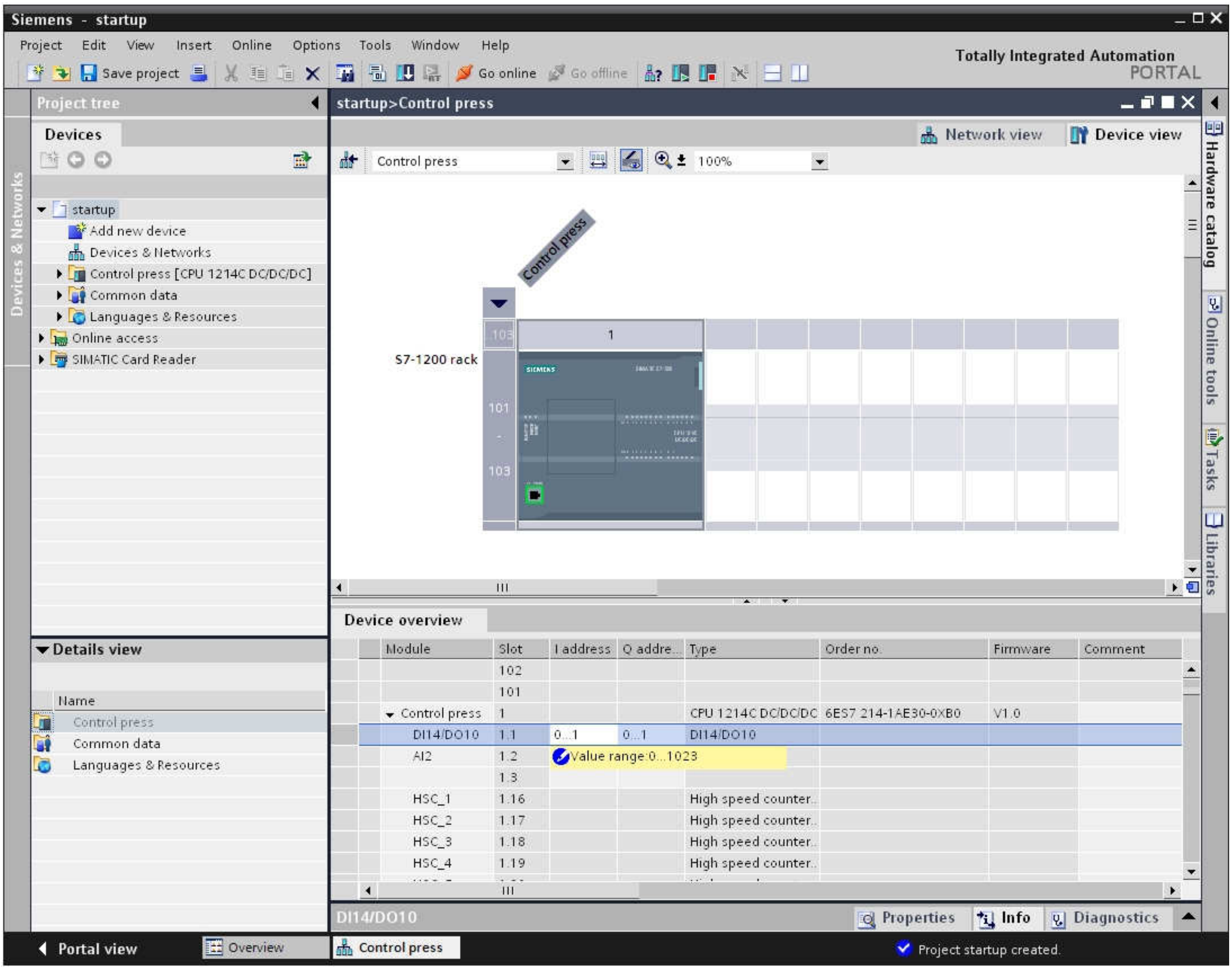


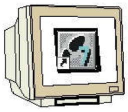


4. Then, we **'Insert a new device'** with the **'Device name Controller Press'**. To this end, we select from the catalog the **'CPU1214C'** with the matching order number (→ Insert new device → Controller press → CPU1214C → 6ES7 ……. → Insert)

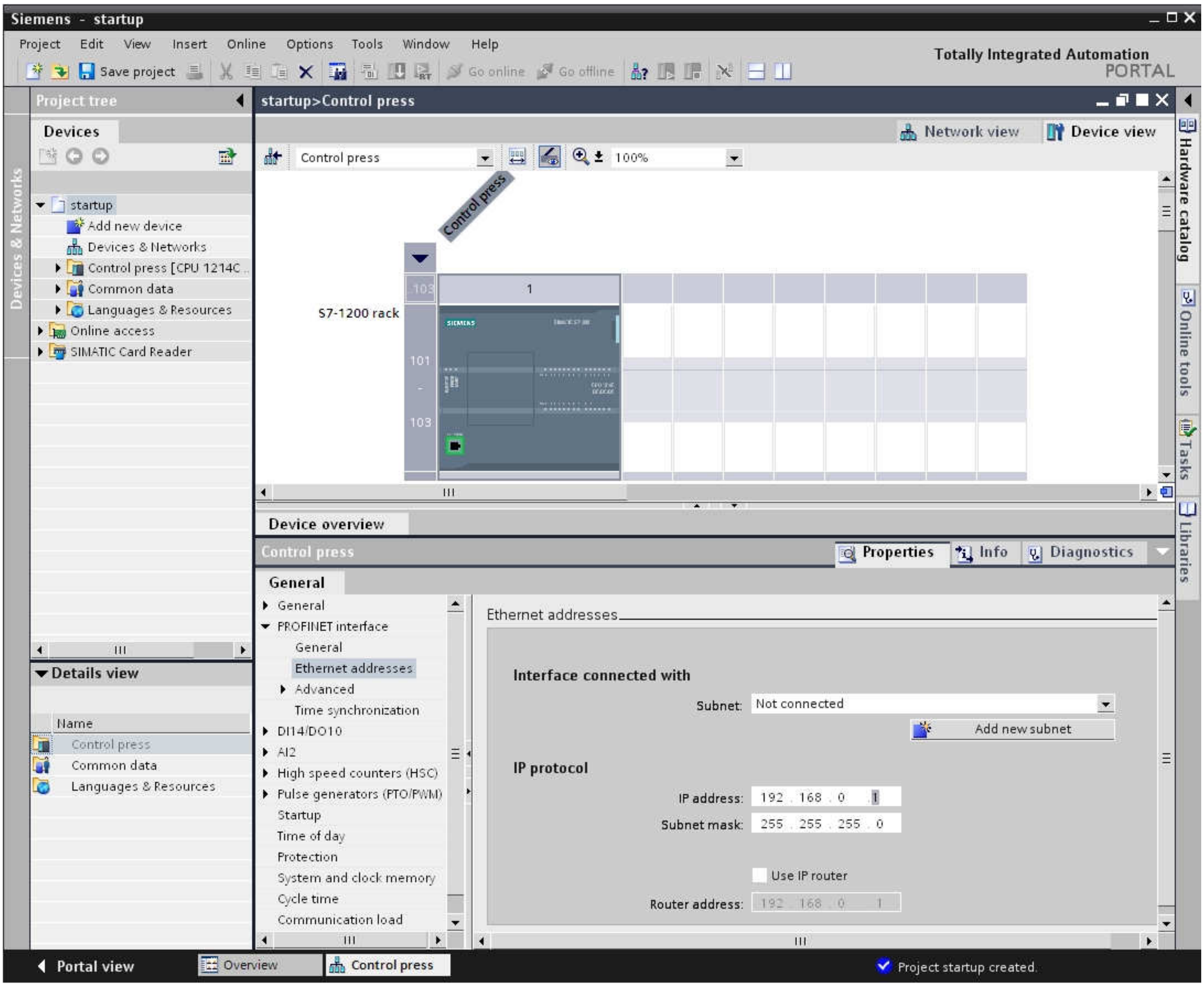


5. Now, the software automatically changes to the project view with the opened hardware configuration. Here, additional modules can be inserted from the hardware catalog (to the right!), and in the **'Device overview'**, the input/output addresses can be set. Here, the integrated inputs of the CPU have the addresses %I0.0 to %I1.5 and the integrated outputs the addresses %Q0.0 to %Q1.1 (→ Device overview → DI14/DO10 → 0…1)





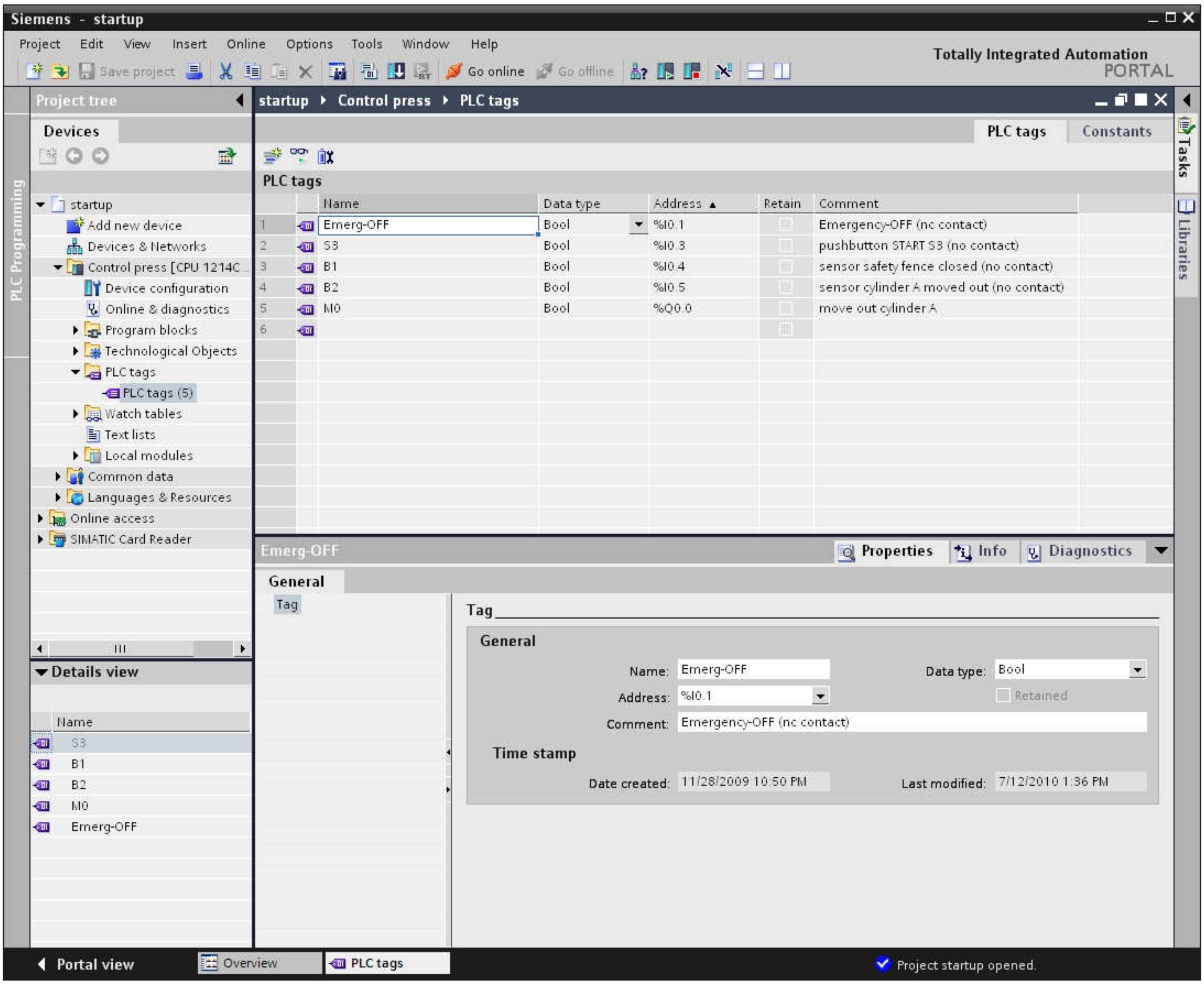
6. So that the software later accesses the correct CPU, its IP address and the subnet screen form have to be set (→ Properties → General → PROFINET interface → IP address: 192.189.0.1 → Subnet screen form: 255.255.255.0)
(refer also to Chapter 3 for setting the programming interface)



7. Since for modern programming, we don’t program with absolute addresses but with variables, we have to specify the **global PLC variables** here.

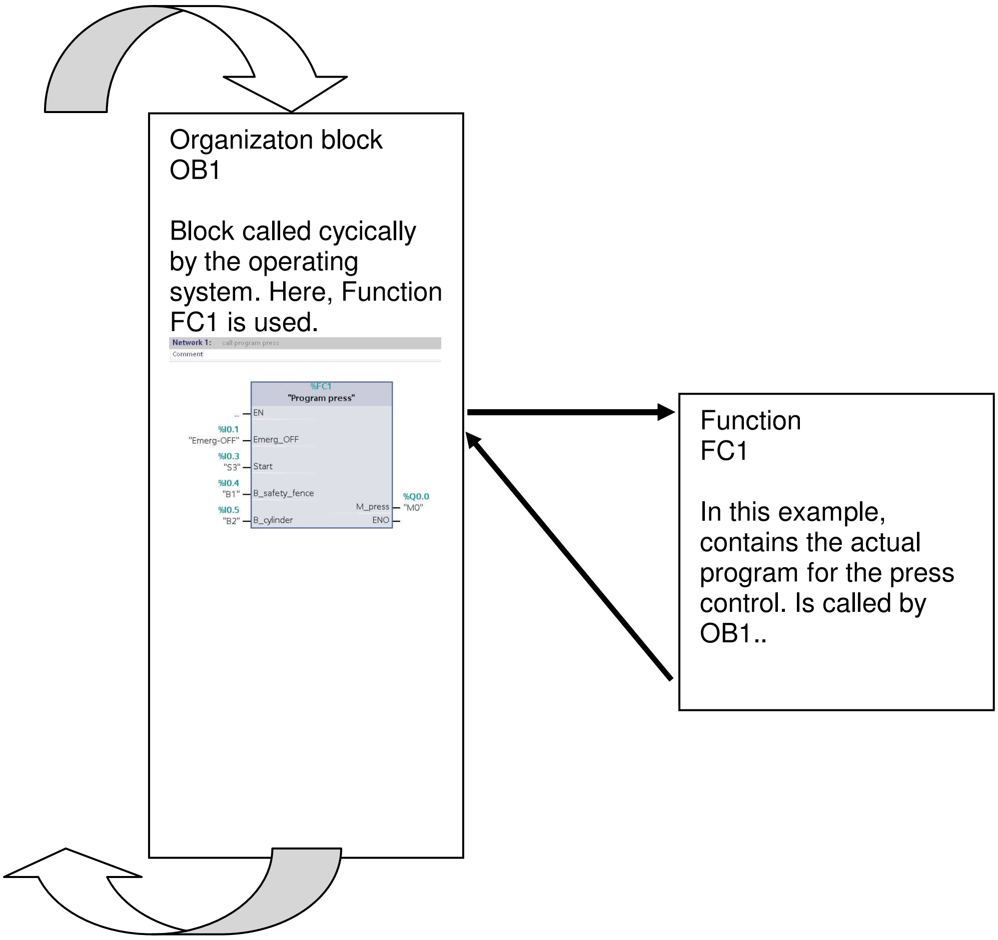
These global PLC variables are descriptive names with comments for those inputs and outputs that are used in the program. Later, during programming, the global PLC variables can be accessed by means of this name.
These global variables can be used in the entire program in all blocks.

To this end, first select in project navigation the **'Controller Press[CPU1214C DC/DC/DC]’** and then **’PLC variables’**. With a double click, open the table **’PLC variables’** and as shown below, enter the names for the inputs and outputs (→ Controller Press[CPU1214C DC/DC/DC]’ → PLC Variables→ PLC Variables)

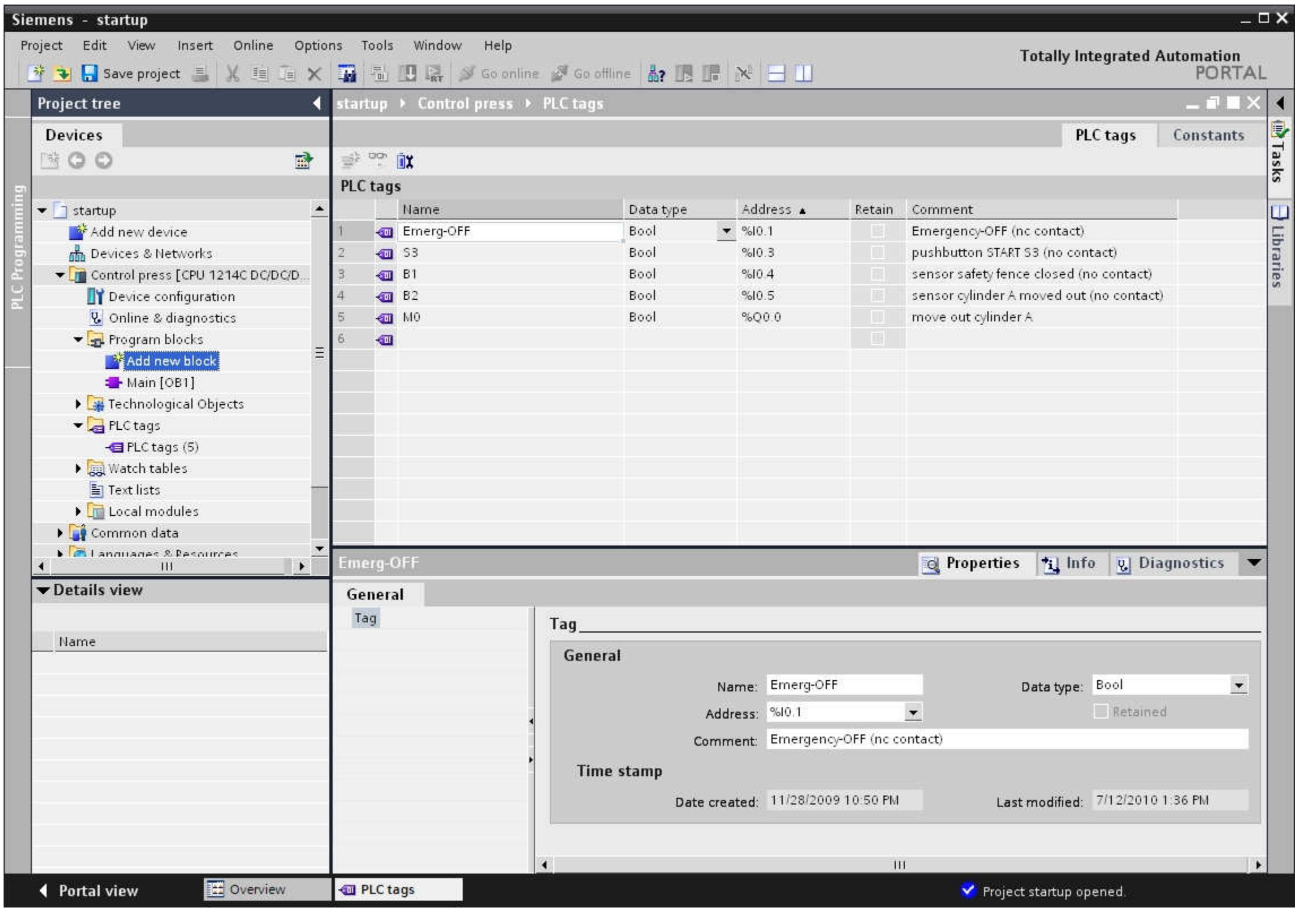


8. The program sequence is written in so-called blocks. As a matter of standard, organization block OB1 already exists.
   It represents the interface to the CPU's operating system, is called by it automatically, and processed cyclically.
   From this organization block, additional blocks can be called in turn for structured programming, such as the function FC1.
   The purpose is to break down an overall task into partial tasks. These can be solved more easily and tested in their functionality.

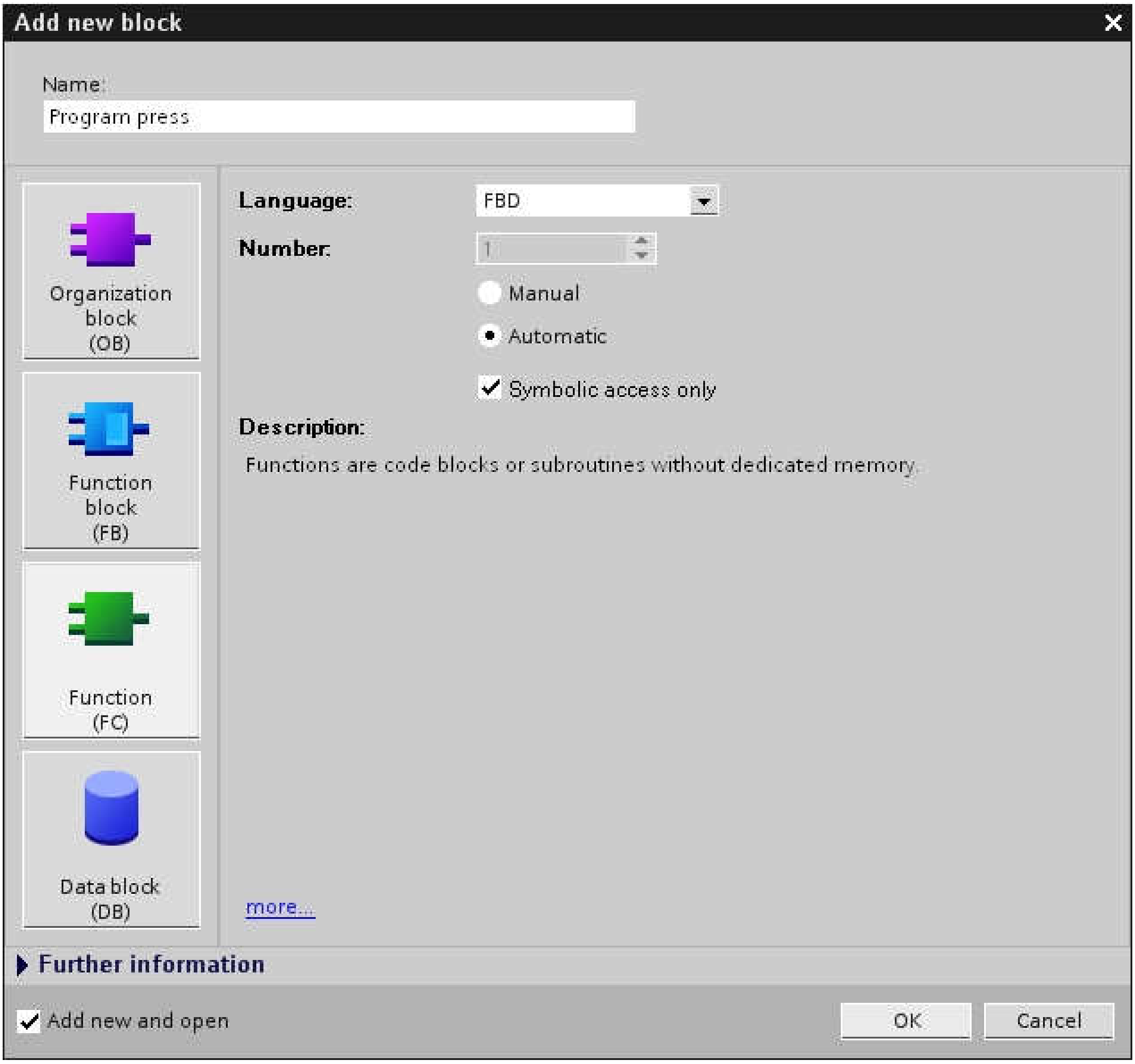
Program structure of the example:



9. To generate the function FC1, in Project Navigation first select **'Controller Press[CPU1214C DC/DC/DC]'** and then **'Program blocks'**. Next, double click on **'Insert new block'** (→ Controller Press[CPU1214C DC/DC/DC]' → Program blocks → Insert new block)



10. In the selection, select ’**Function (FC)**’ and assign the name ’**Program press**’. As programming language, '**FBD**’ is entered. Enumeration is automatic. Since this FC1 is called later with the symbolic name anyhow, the number is no longer that important. Accept your input with ’**OK**’. (→ Function (FC) → Program Press → FBD → OK)



11. The block **’Program Press[FC1]’** will be opened automatically. However, before the program can be written, the block’s interface has to be declared.
When the interface is declared, the local variables known only in this block are specified.

The variables consist of two groups

• Block parameters that generate the interface of the block for the call in the program.

Type	Name	Function	Available in
Input parameters	Input	Parameters whose values the block reads	Functions, function blocks and some types of organization blocks
Output parameters	Output	Parameters whose values the block writes	Functions and function blocks
InOut parameters	InOut	Parameters whose value the block reads when it is called and after processing, writes again to the same parameter	Functions and function blocks

• Local data that is used for storing intermediate results.

Type	Name	Function	Available in
Temporary local data	Temp	Variables that are used for storing temporary intermediate results. Temporary data is retained for one cycle only.	Functions, function blocks and organization blocks
Static local data	Static	Variables that are used for storing static intermediate results in the instance data block. Static data is retained also over several cycles, until it is written anew.	Function blocks

12. When declaring the local variables, in our example the following variables are needed.

Input:

EMERGENCY_OFF	Here, EMERGENCY OFF monitoring is entered
Start	Here, the start button is entered
B_Screen	Here, the status of the protection screen is entered
B_Cylinder	Here, the status of the sensor Cylinder Extended is entered

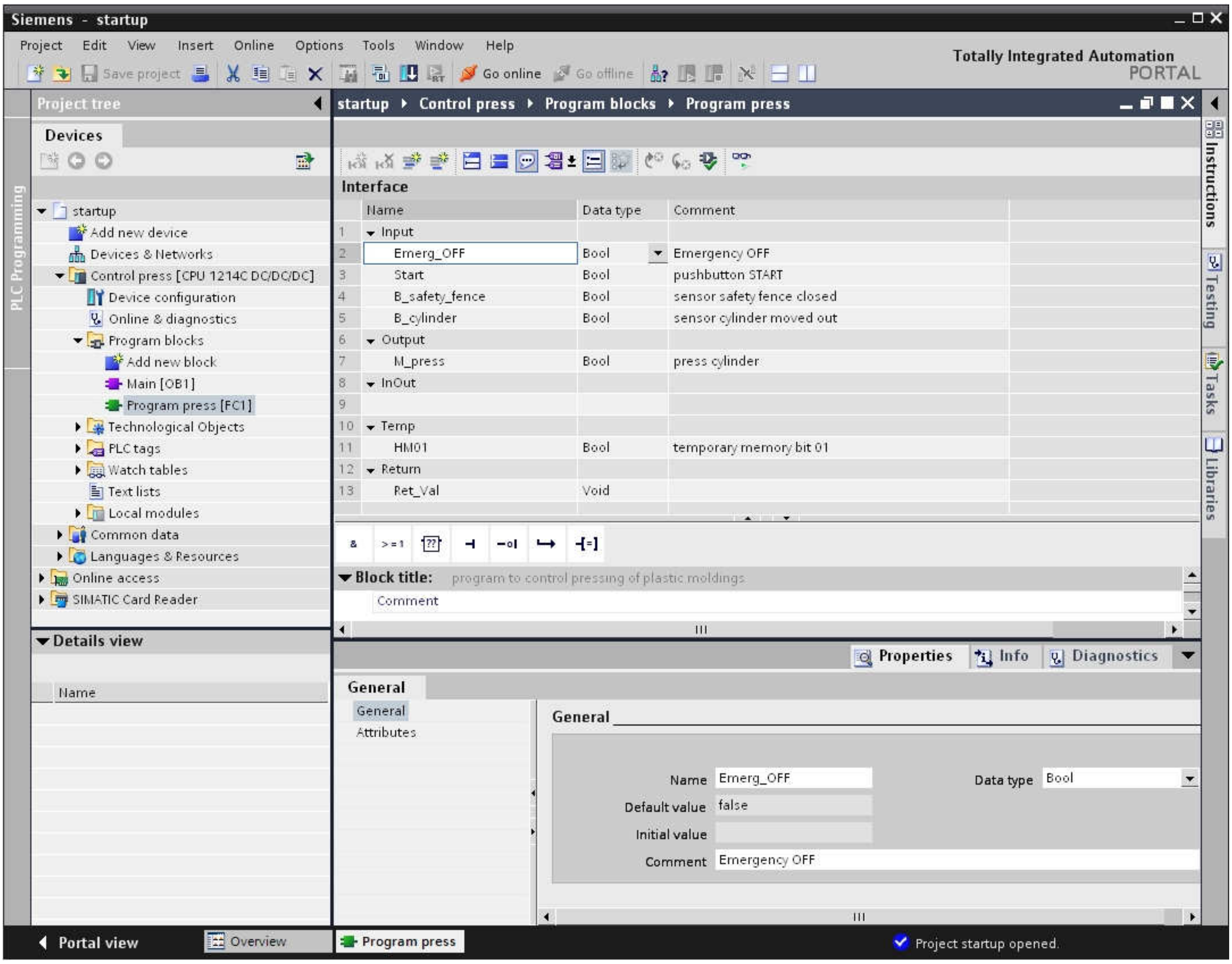
Output:

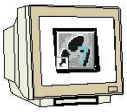
M_Press	Here, a status for the output Press Cylinder is written

Temp:

HM01	Auxiliary flag 01 for the SR FlipFlop

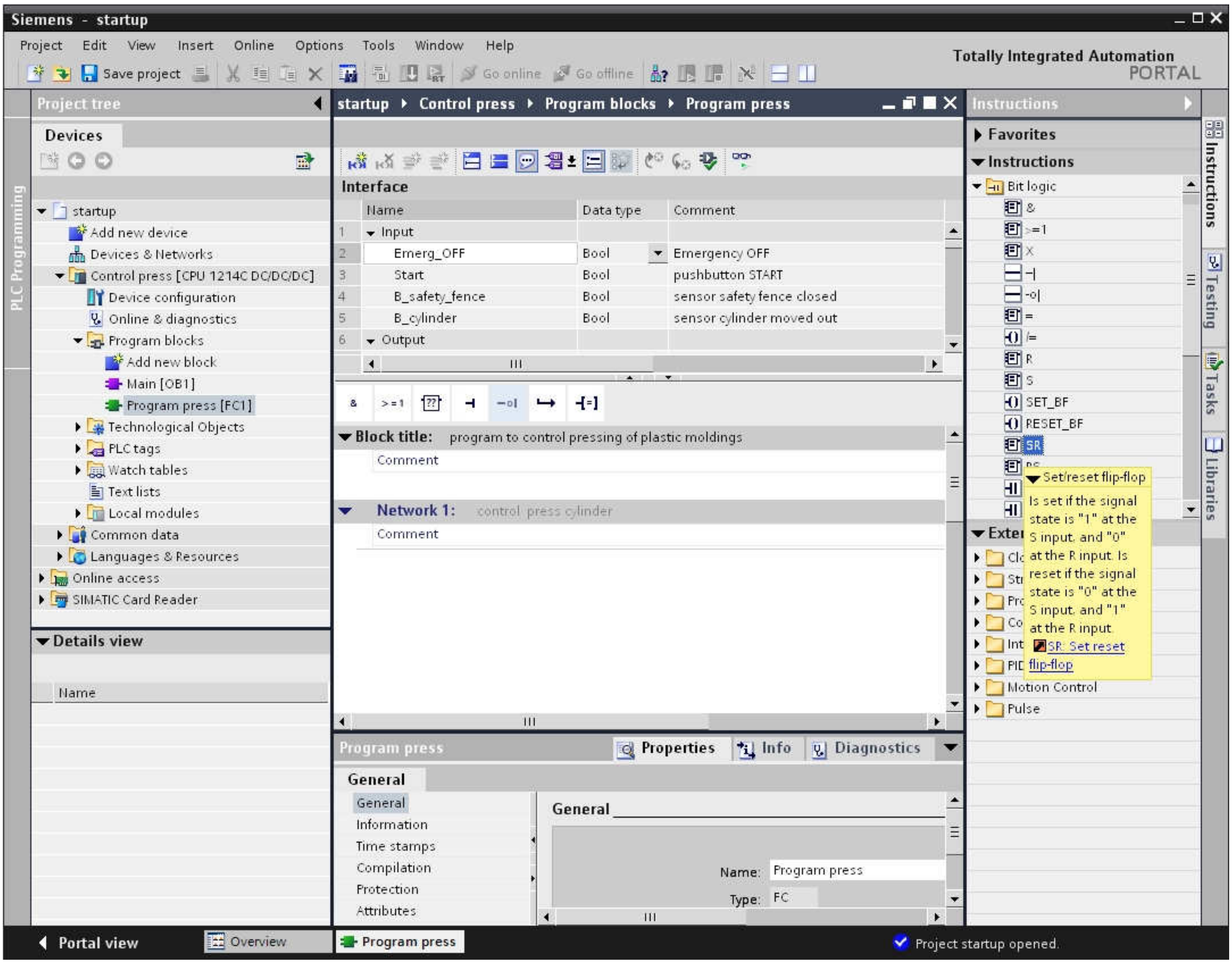
All variables in this case are of the type 'Bool'; which means variables that only can have the status '0' (false) or '1' (true).
To make it easier to follow them, all local variables should also be provided with a sufficient comment.



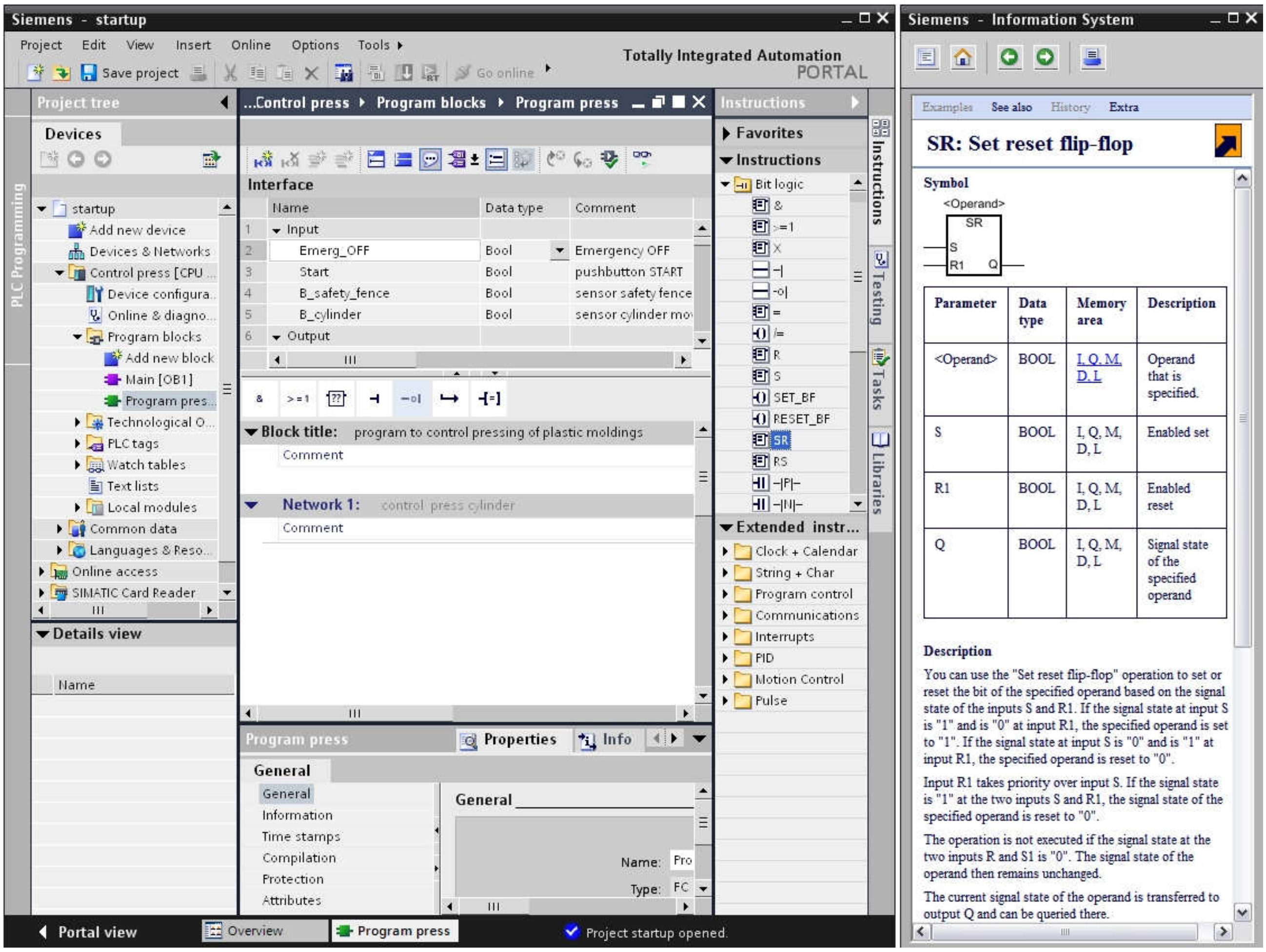


13. After having declared the local variables, we can now start programming. To provide a better overview, we program in networks. A new network can be inserted by clicking on the symbol  **'Insert network'**. Like the block itself, each network should be documented in the title line. If a longer text is needed for the description, the **'Comment'** field can be used (→ )

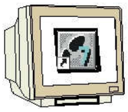
To generate our solution, we need an **'SR Flipflop'**. It is located under **'Instructions'** in the folder **'Bit combinations'**. If you point with the mouse to an object such as the SR flipflop, detail information about this object will be displayed (→ Instructions → Bit combination → SR)



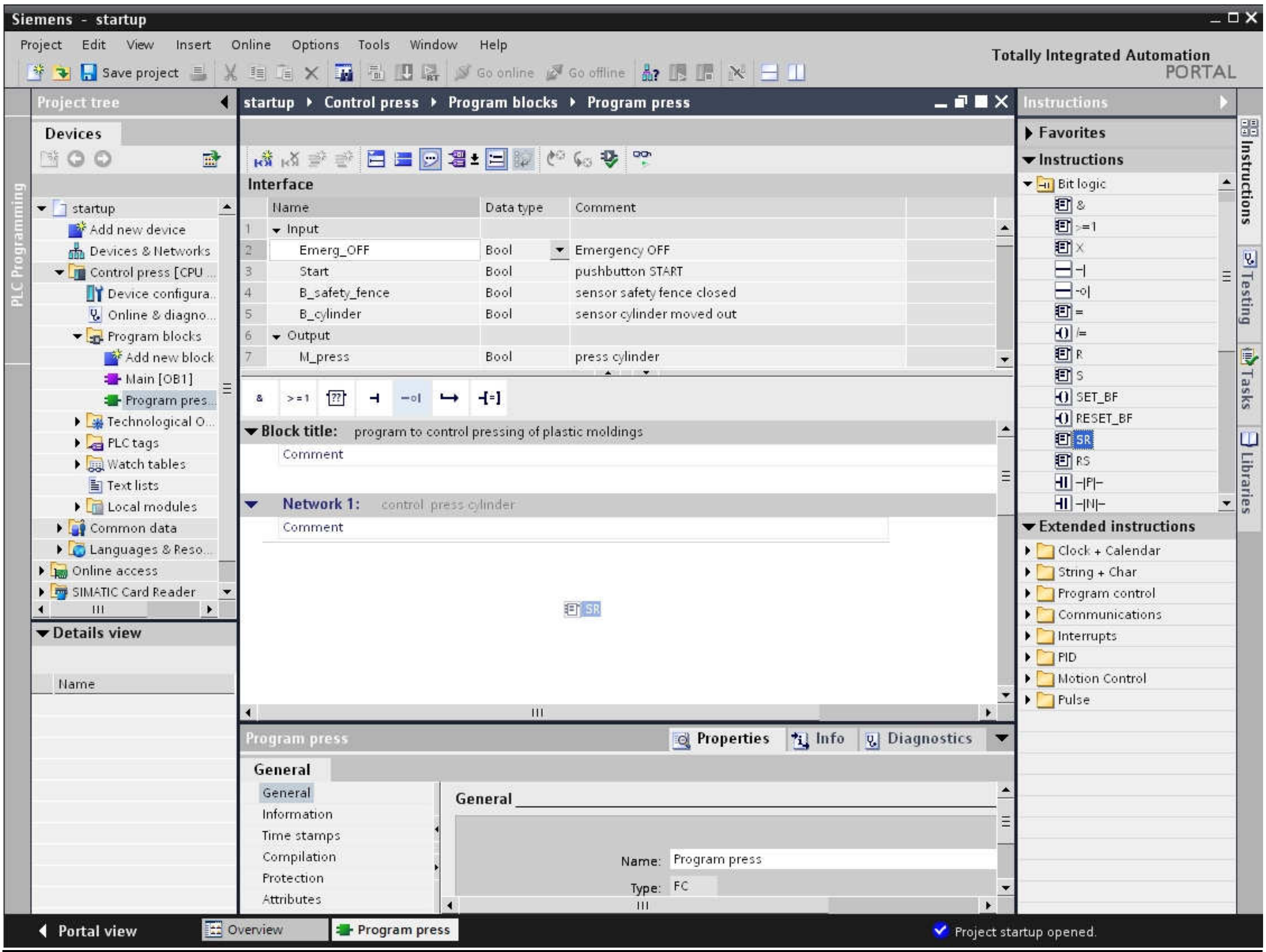
14. If you highlight an object and then press **'F1'** on your PC, you will be provided with online help about this object in a window to the right (→ F1)



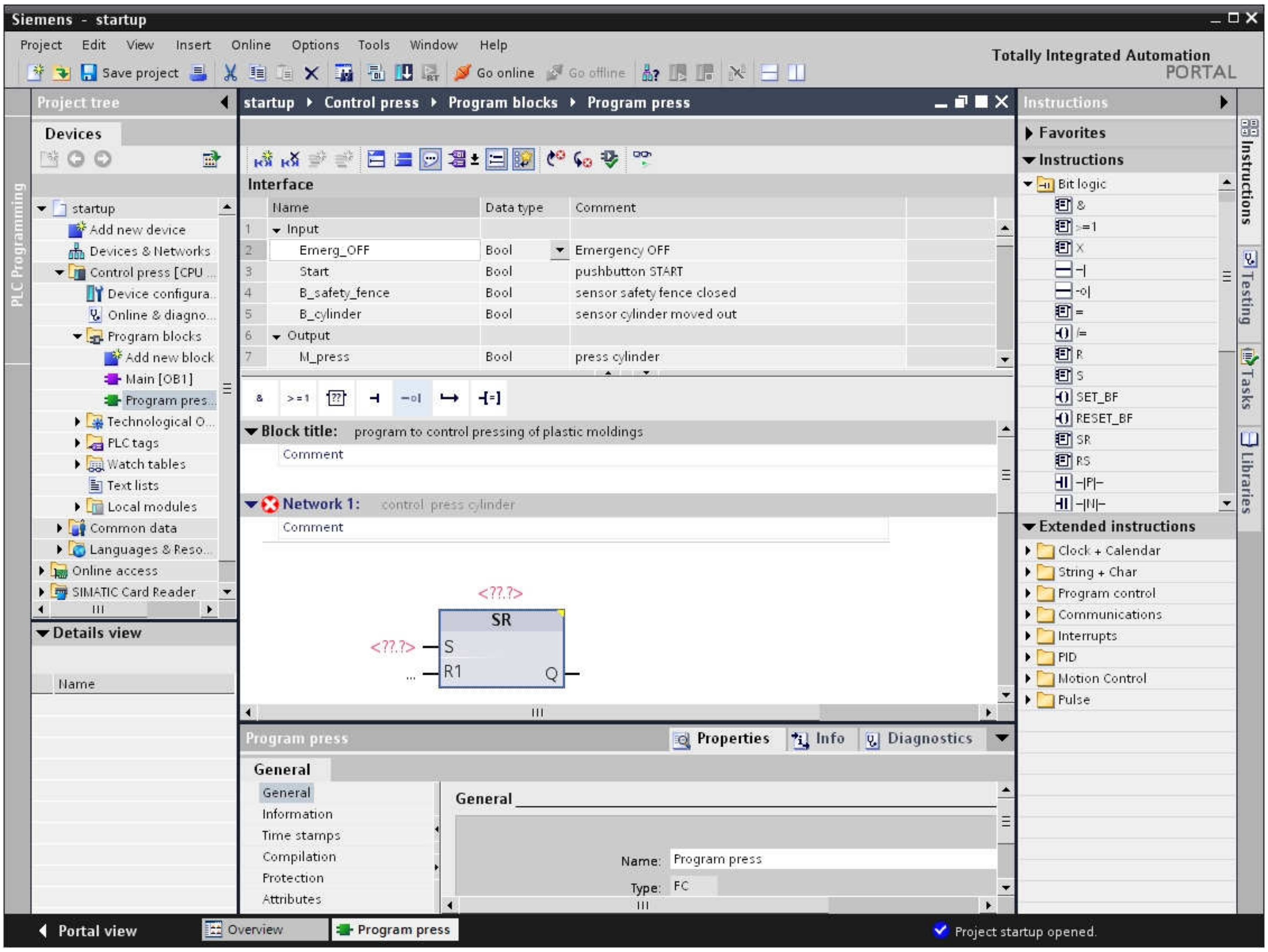
Note: Here, extensive information is provided in online help regarding the function and the wiring of the SR flipflop.



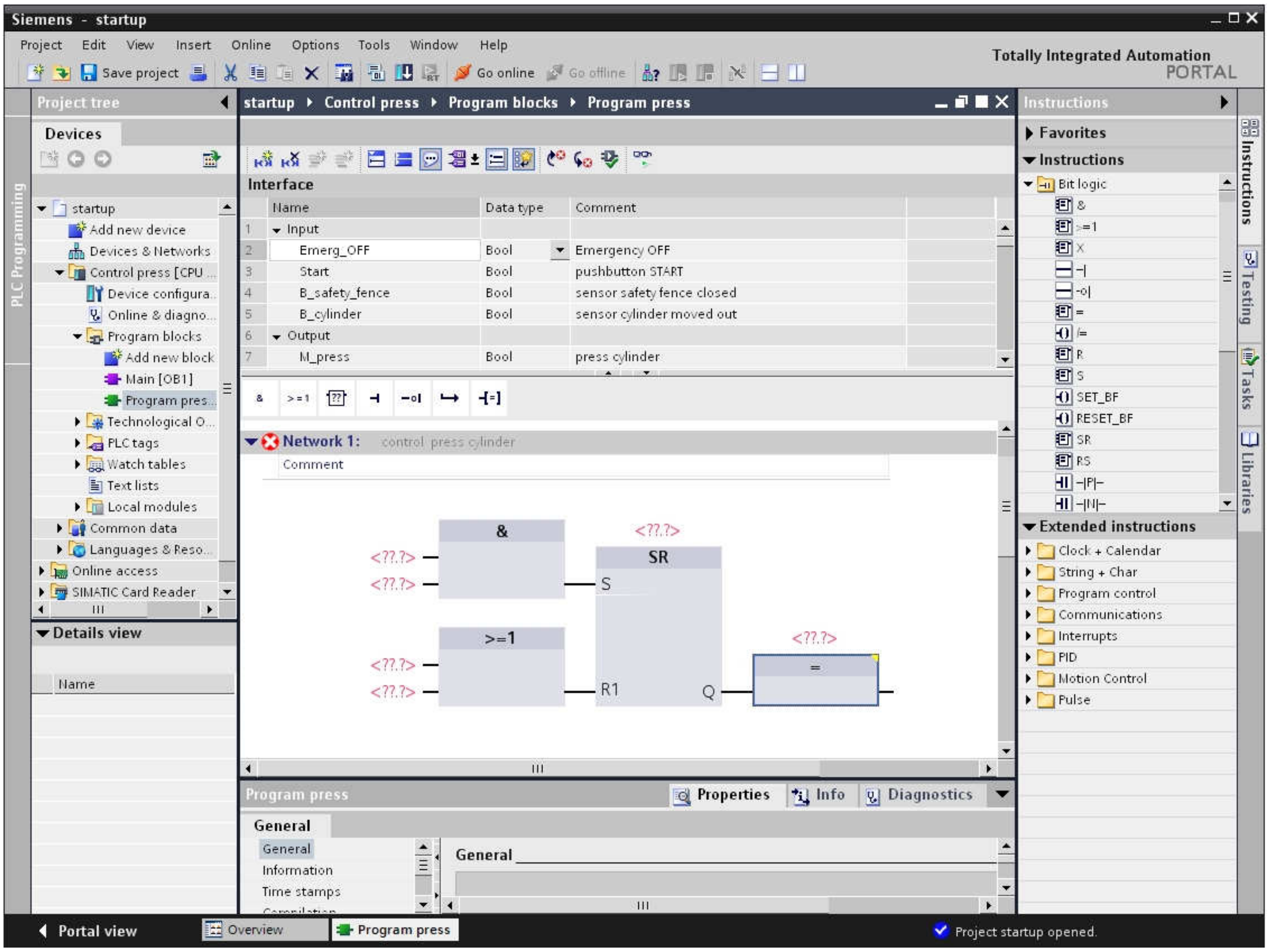
15. Now, drag the SR flipflop with the mouse below the comment in Network 1. (→ SR)

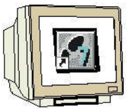


16. Next, highlight the **S**et input of the SR flipflop and click on AND in the favorites (→ S → Favorites → AND)

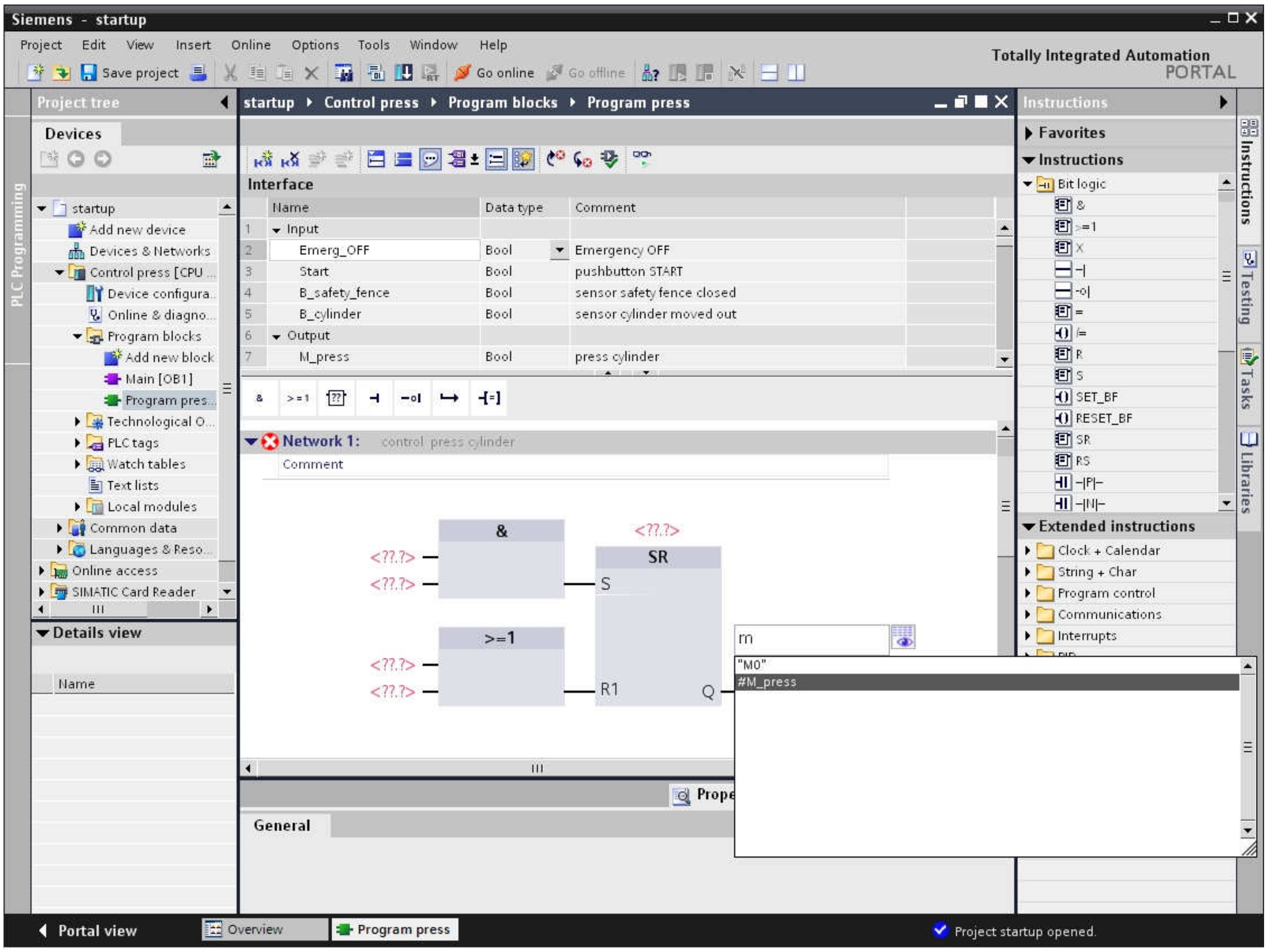


17. Exactly in the same way we place the >=1 OR at the **R1** input and the -[=] assignment at the **Q** output of the SR flipflop (→ R1 → >=1 OR → Q → -[=] assignment)

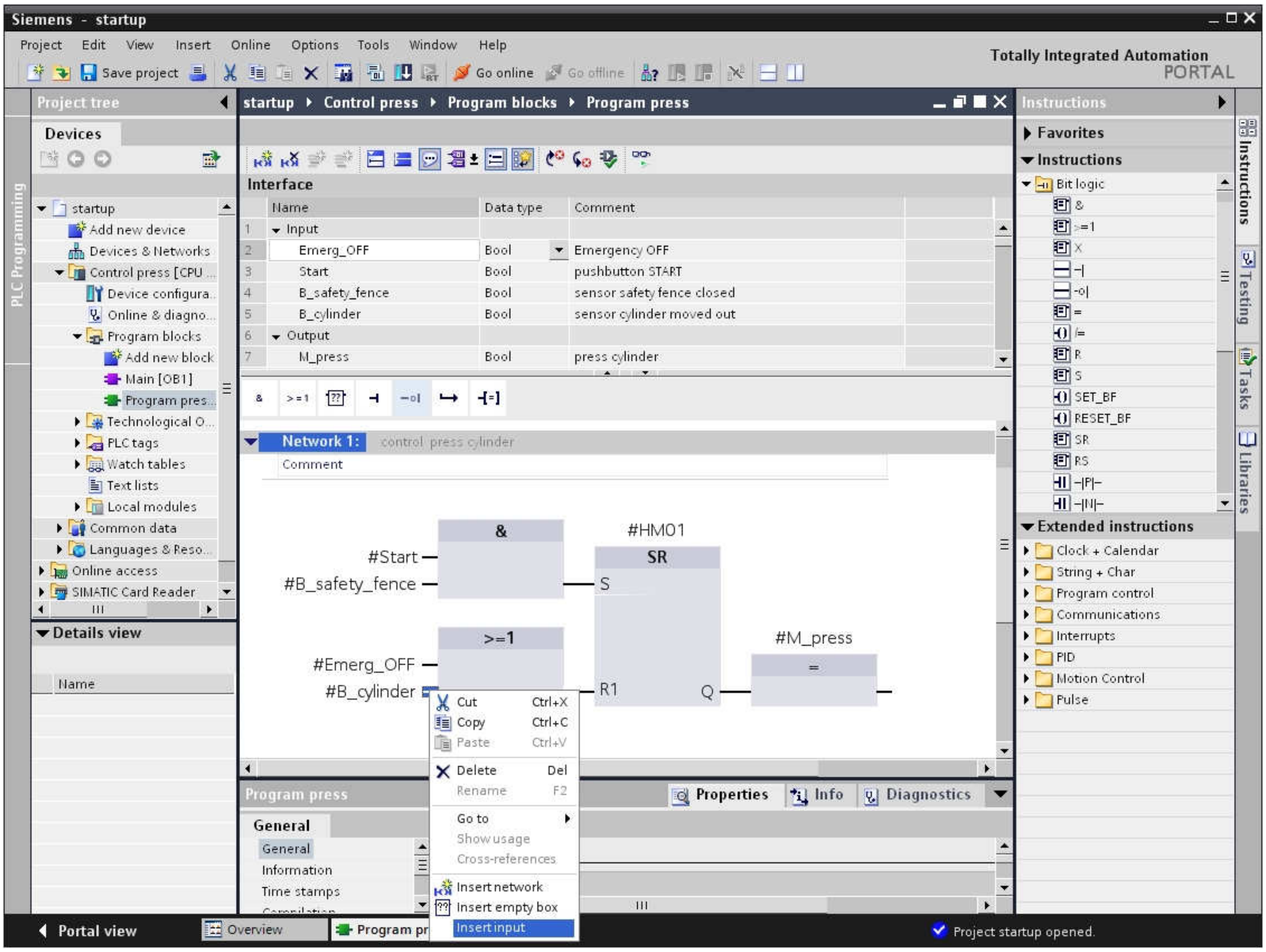


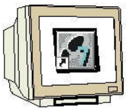


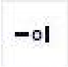
18. Now, we enter the local variables. It suffices to enter the first letter of the local variables in the fields at the commands. Then we can select the desired variable from a list. Local variables are always identified with the symbol **'#'** preceding the name (→ #M_Press).

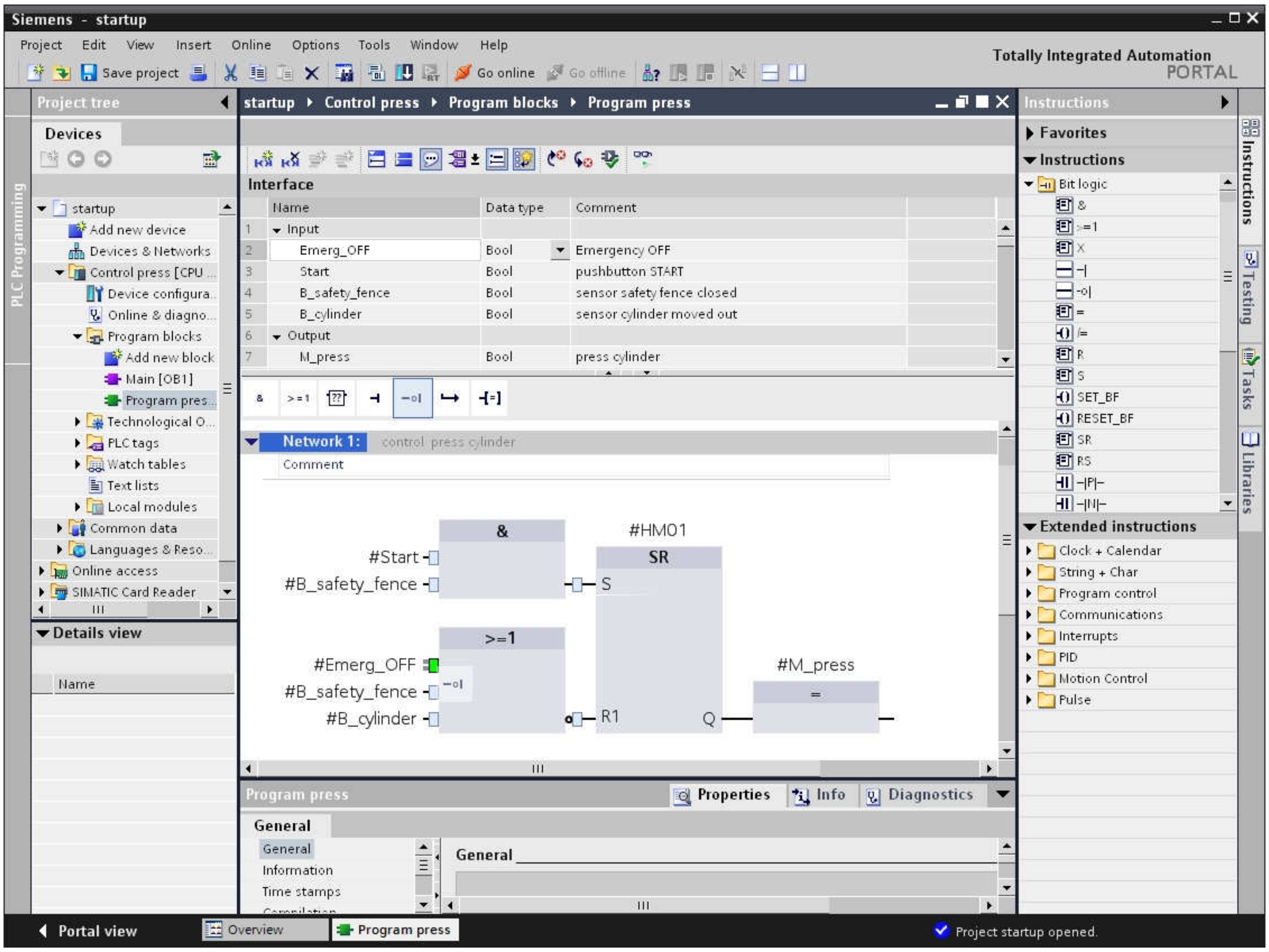


19. Likewise, add the other local variables. At the OR, another input is to be inserted. To this end, highlight the lowest input with the right mouse key and select **'Insert input'** (→ Insert input)

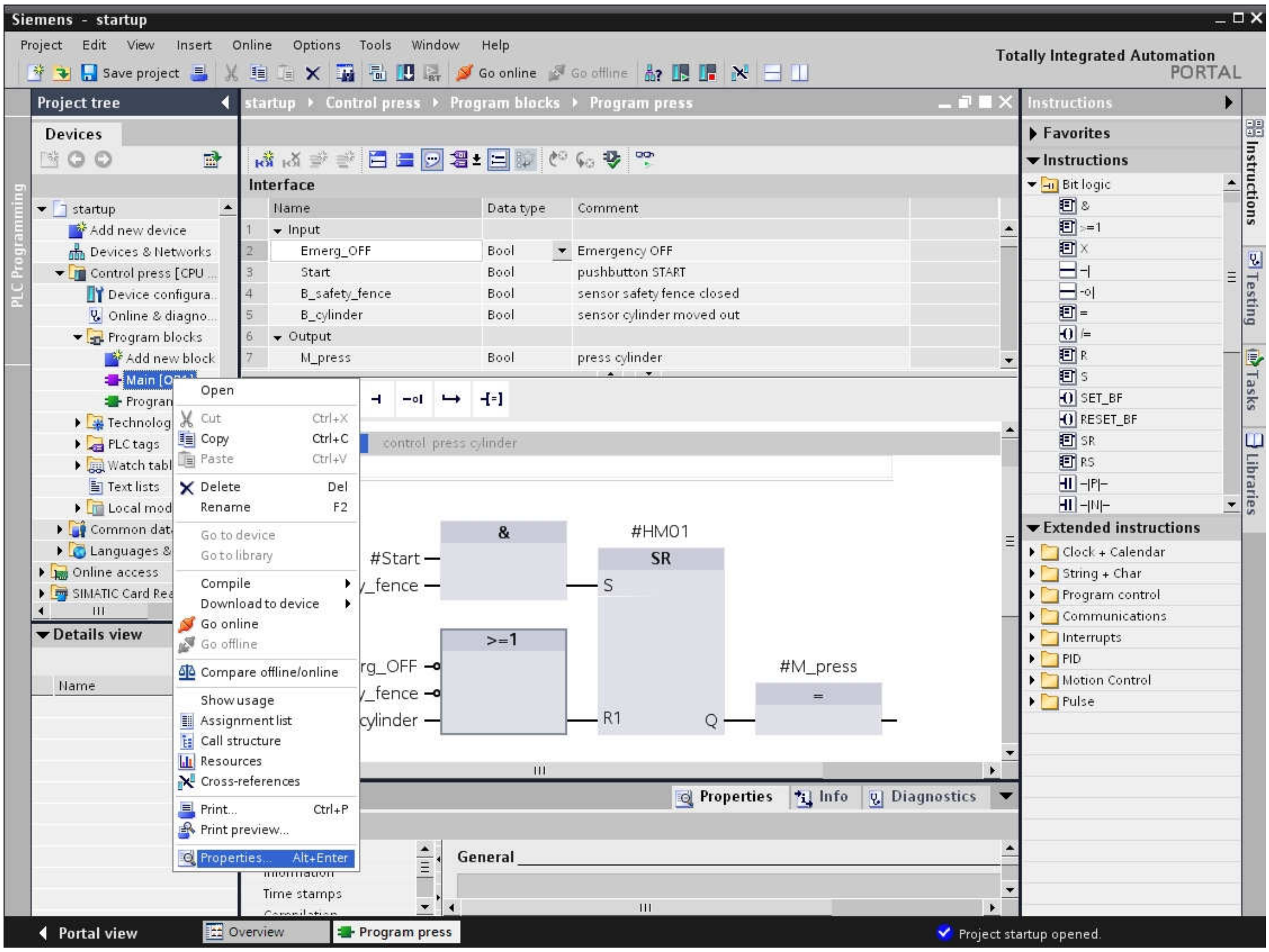


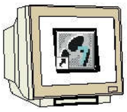


20. Assign the local variable shown here to this input also. If an input is to be inverted, simply drag the symbol  Negation from the **'Favorites'** to the input. (→ Favorites → )

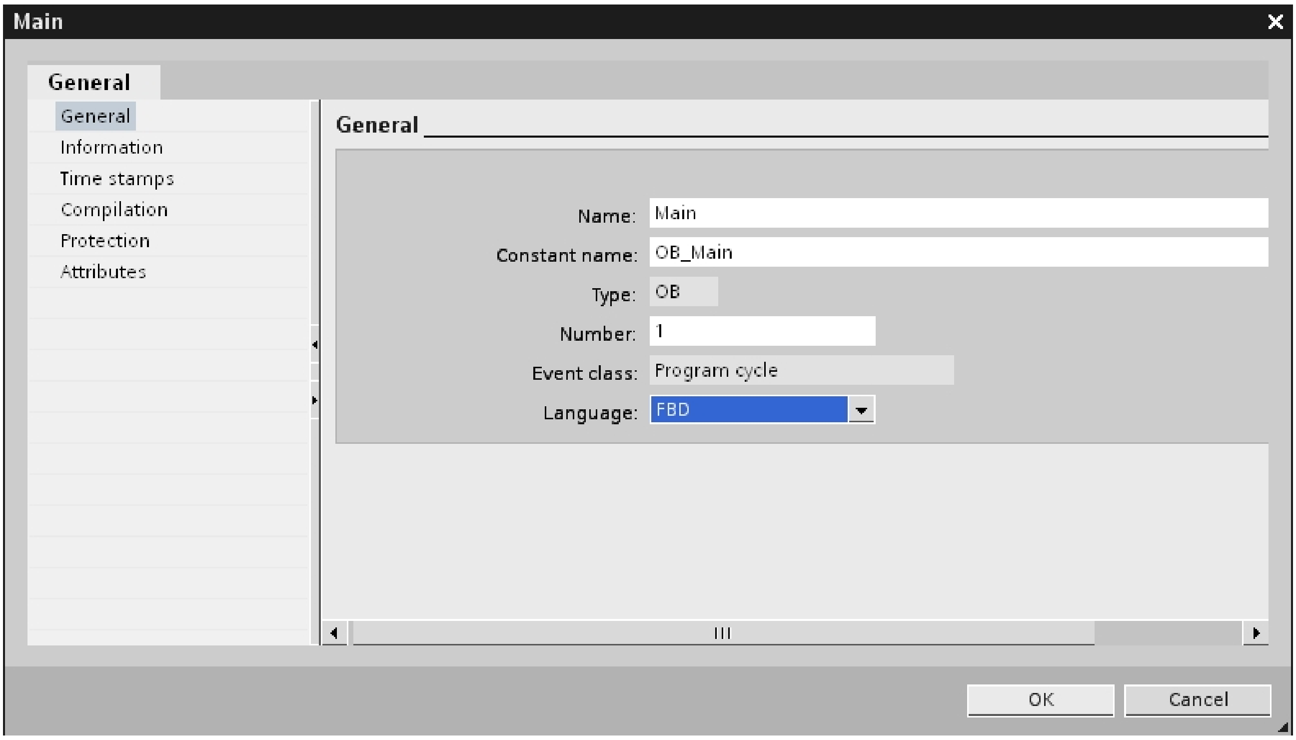


21. Next, the **’Properties’** of the cyclically processed block **’Main[OB1]’** are selected. Block properties can be modified (→ Properties → Main[OB1])

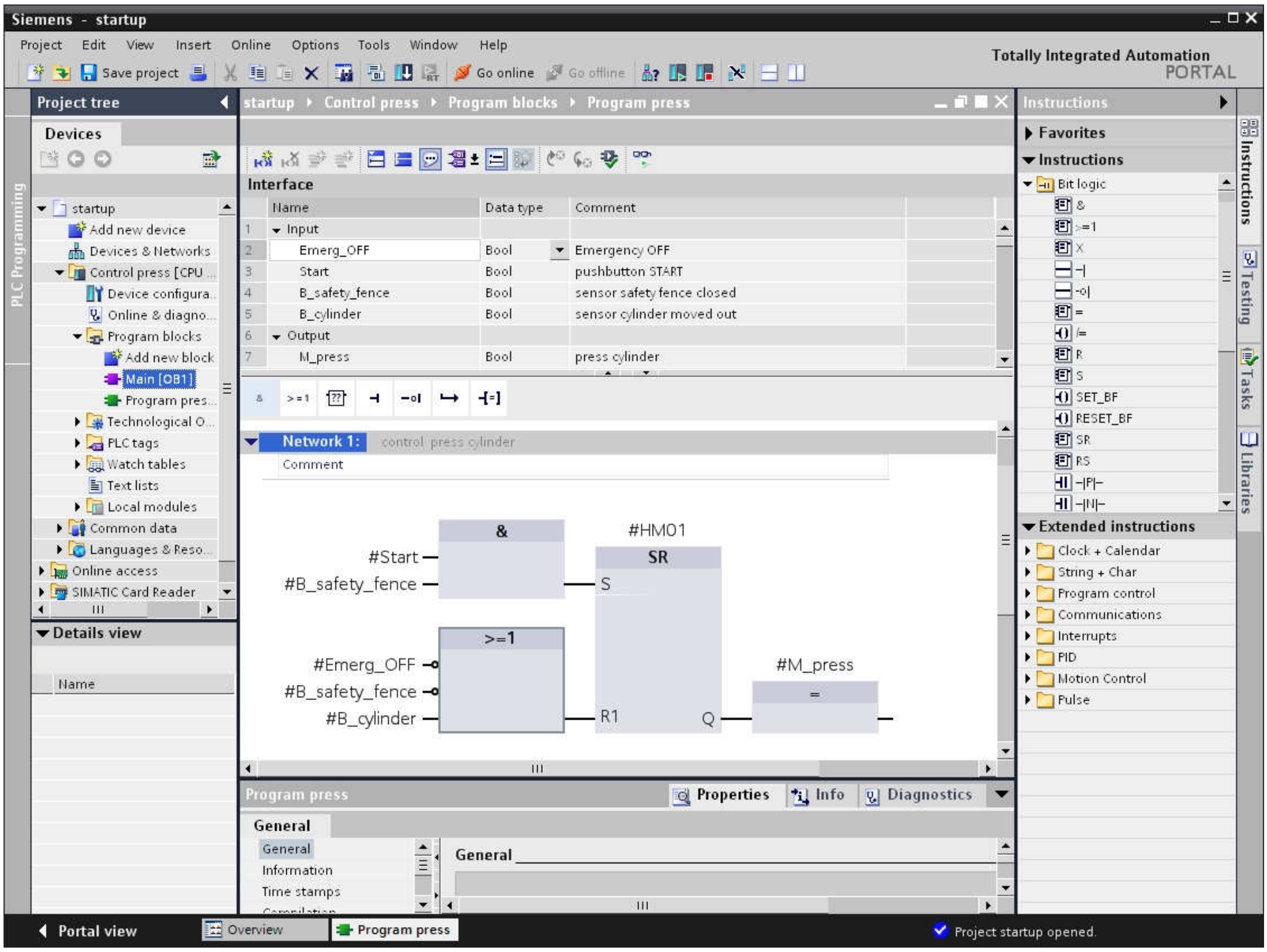


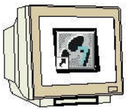


22. In the properties, select the programming-’**Language**’ function block diagram ’**FBD**’. (→ FBD → OK)

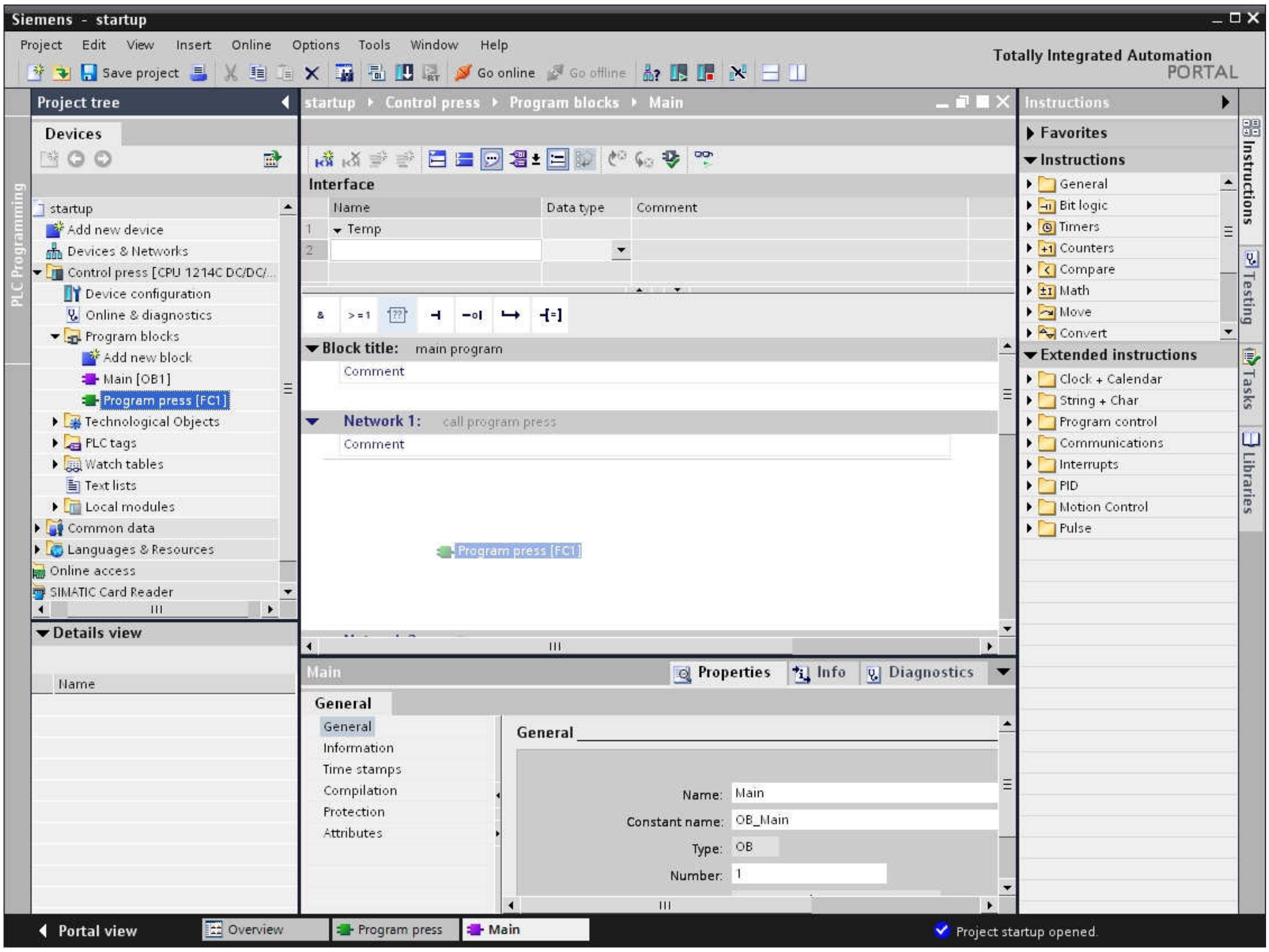


23. As mentioned previously, the block "Program Press“ has to be called from the program block Main[OB1]. Otherwise, the block would not be processed al all. Open this block by double clicking on **’Main[OB1]’** (→ Main[OB1])

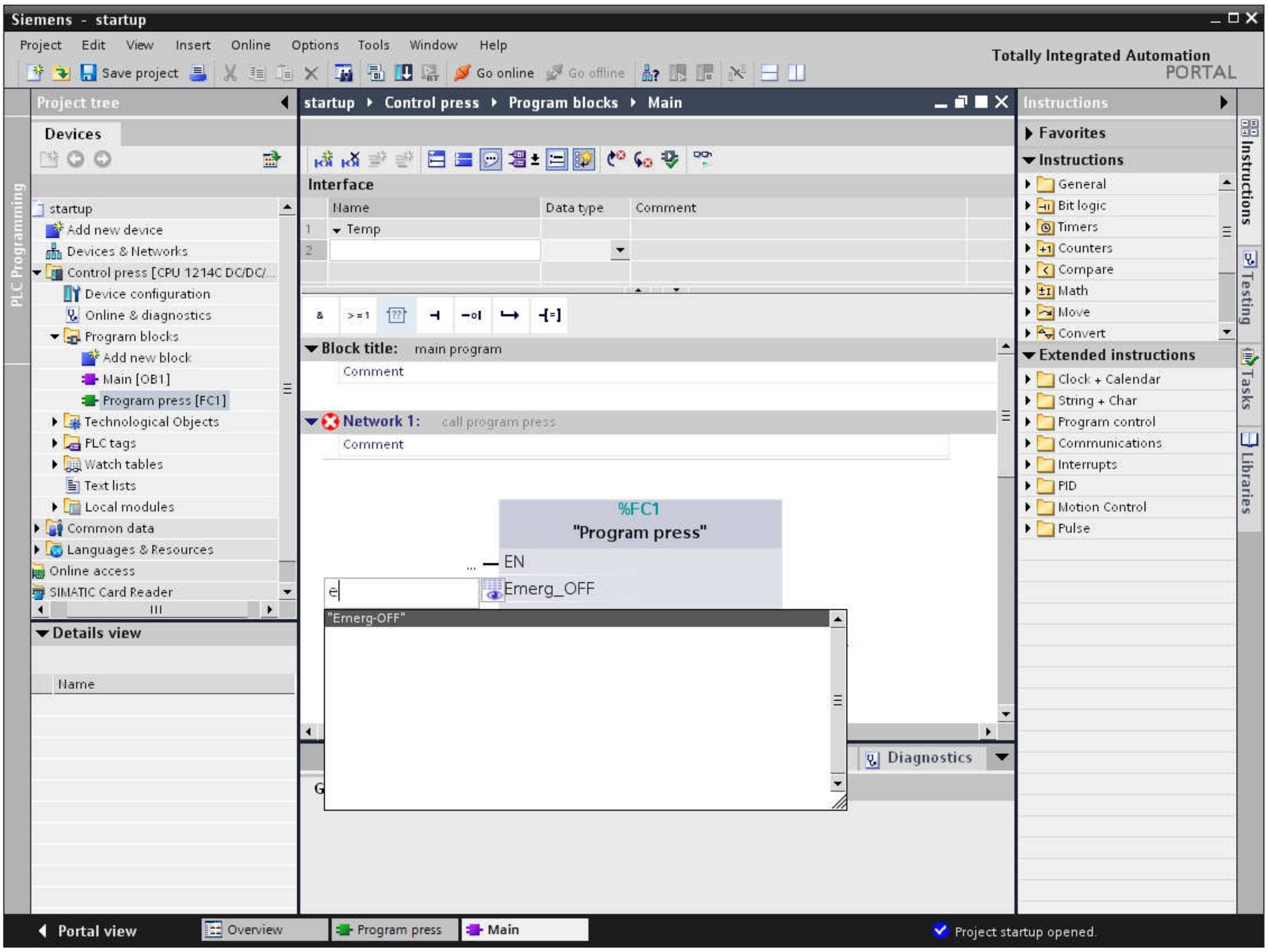




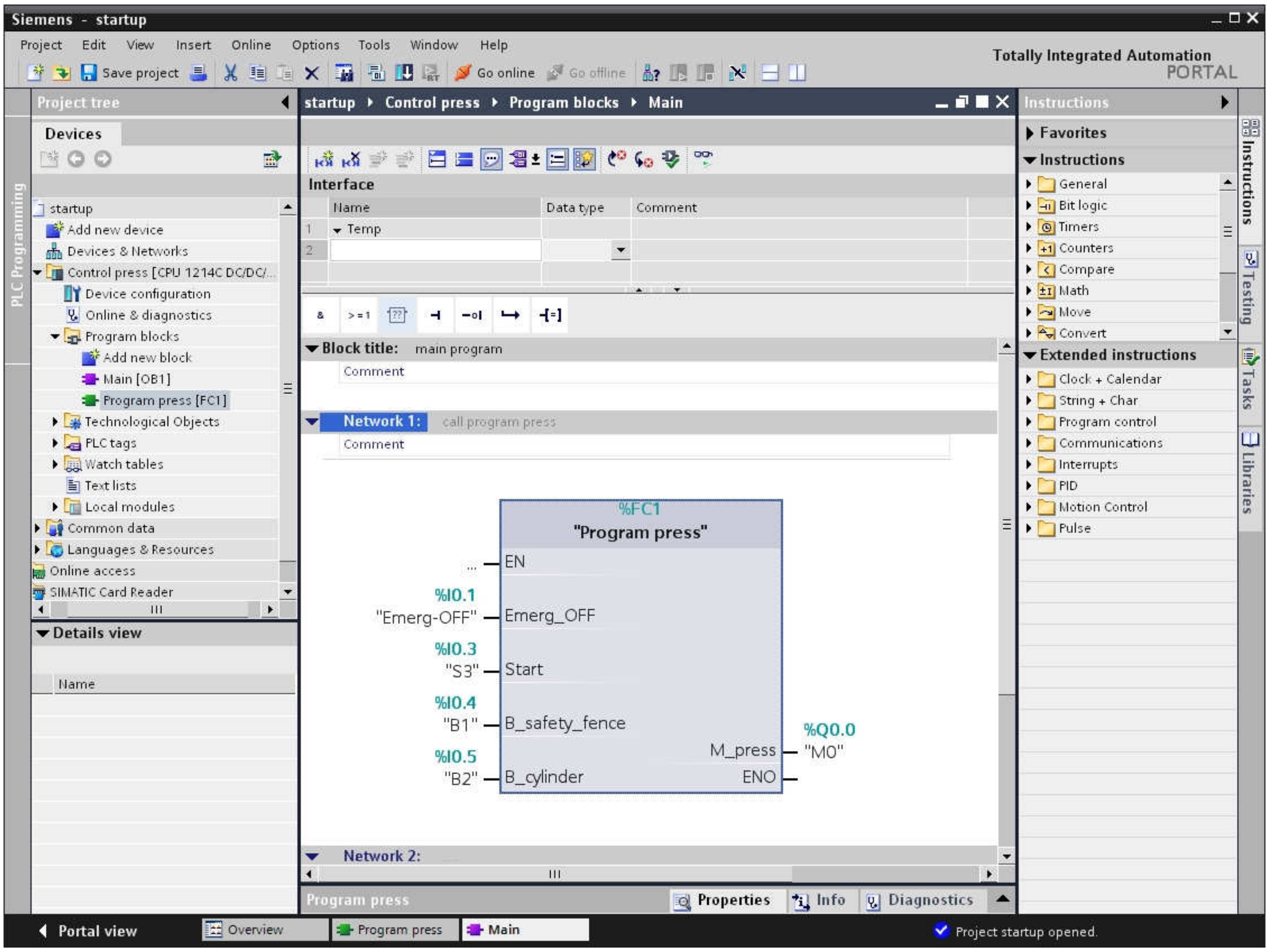
24. The block **"Program Press“** can then simply be dragged with Drag&Drop to Network 1 of the block Main[OB1]. Don’t forget to document the networks also in block Main[OB1] (→ Program Press)

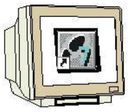


25. Next, the interface parameters of the block "Program Press“ have to be connected to global PLC variables. It is sufficient to enter the first letter of the global variable in the field before the local variable of the block. Then, the desired operand can be selected from a list (→ "EMERGENCY OFF“)

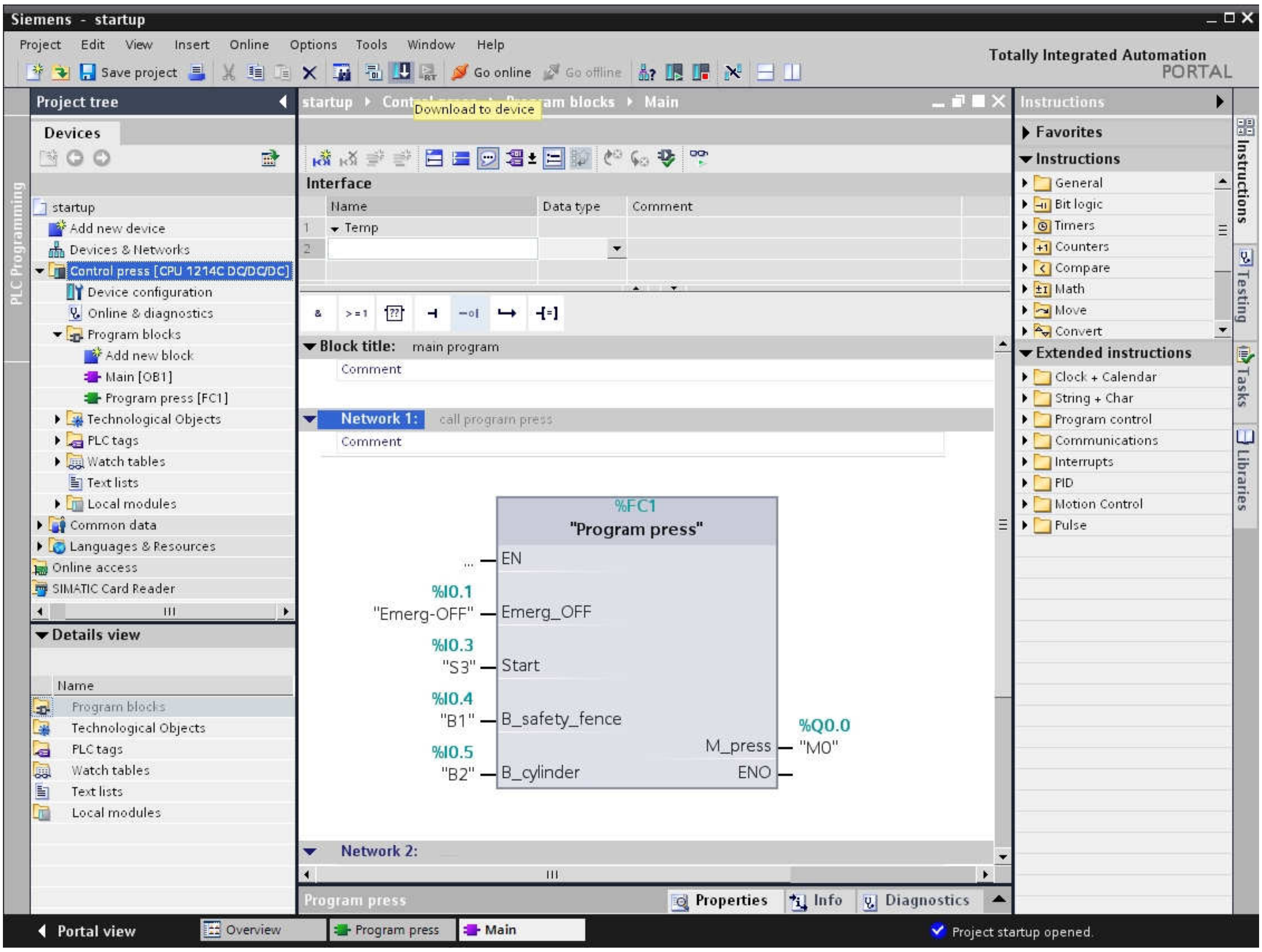


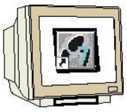
26. Likewise, connect the input variables **’Start’**, **’B_Screen’** and **’B_Cylinder’** as well as the output variable **’M_Press’** to the PLC variables shown here. With a mouse click on Store project Save project the project will be stored. (→“S3“ → "B1“ → "B2“ → "M0“ → Save project)



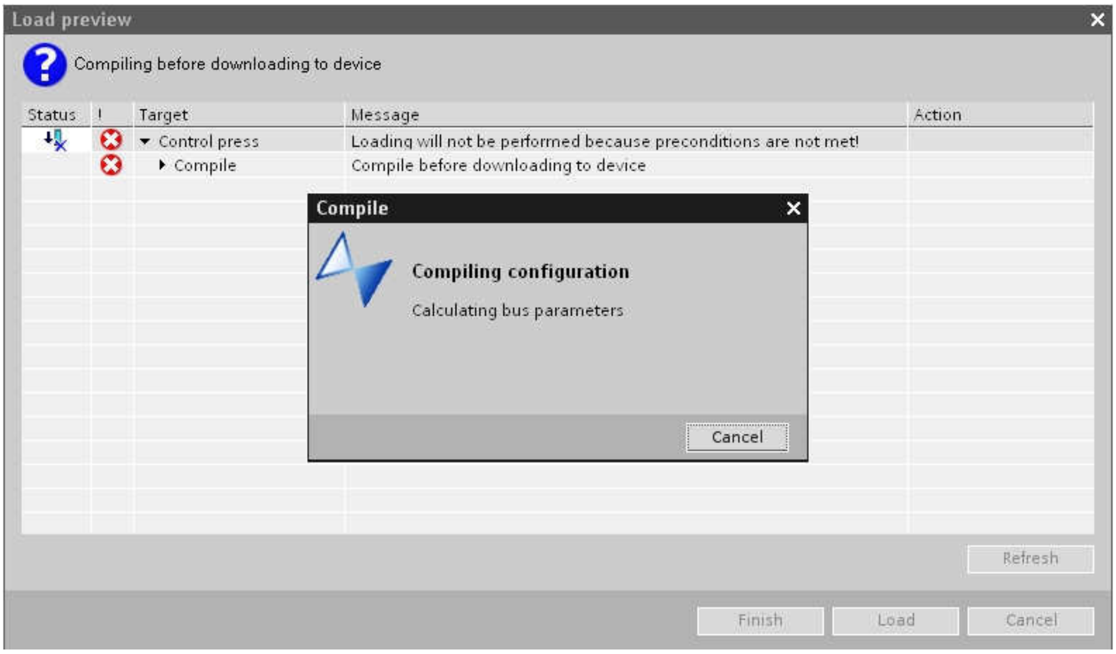


27. To load your entire program into the CPU, first highlight the folder **’Controller Press’** and then click on the symbol Load to device (→ Controller Press →)

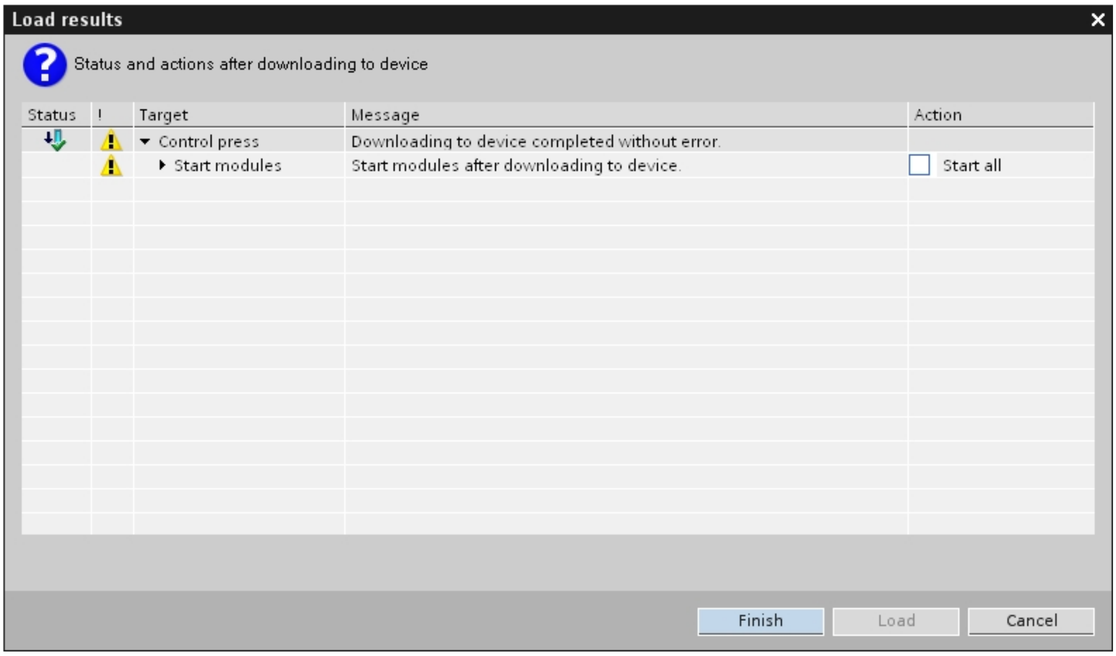




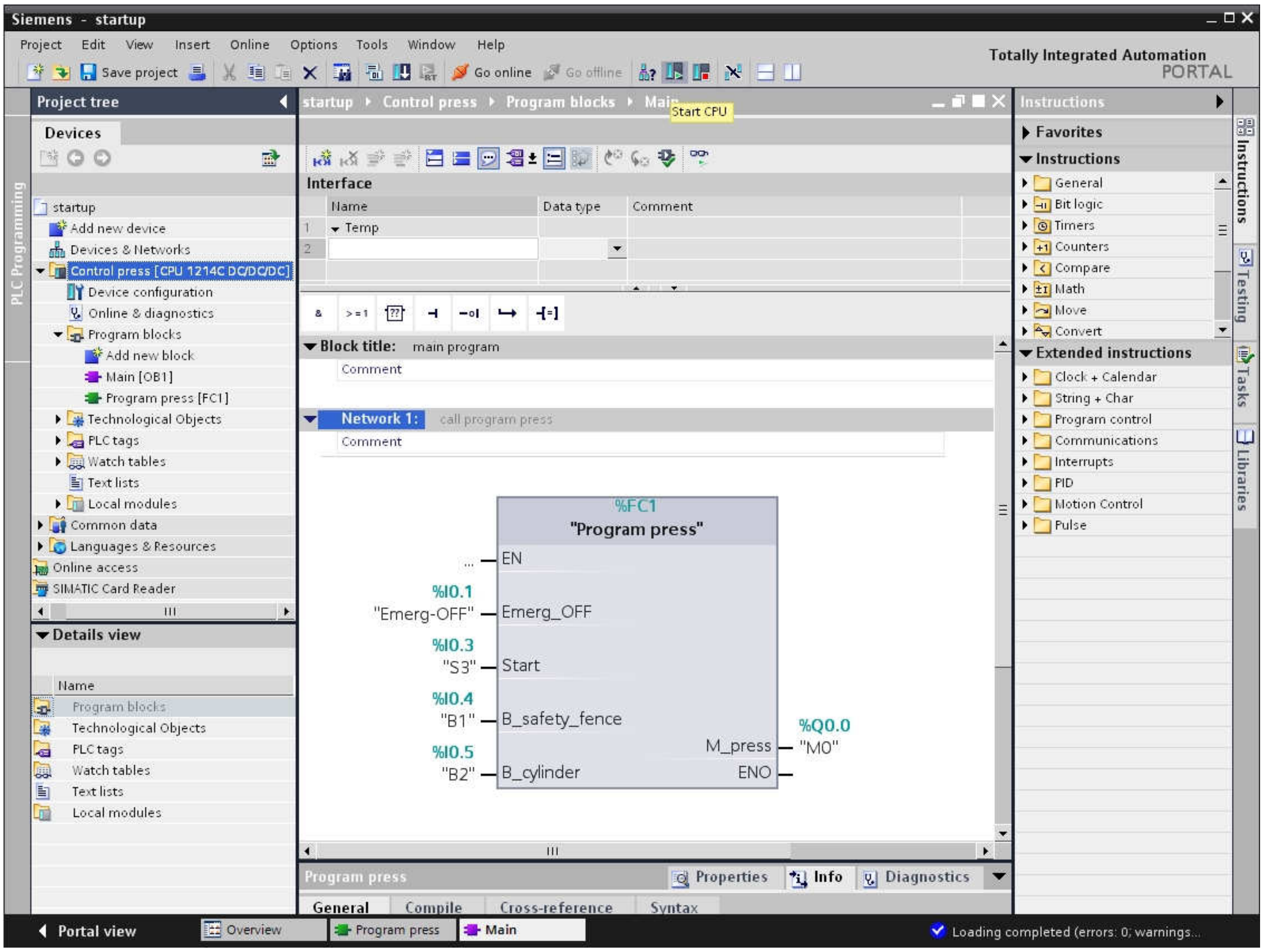
28. During loading, the status is displayed in a window.



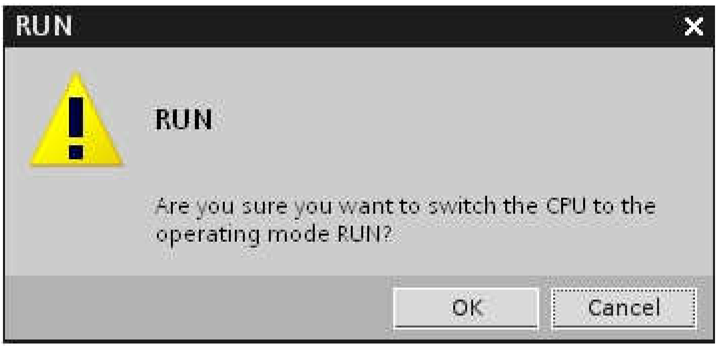
29. If loading was successful, it is displayed in a window. Now click on ’**Complete**’ (→ Complete)

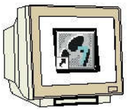


30. Now, start the CPU by clicking on the symbol (→)



31. With **'OK'**, confirm the question whether you actually want to start the CPU (→ OK)





32. By clicking on the symbol  Monitoring on/off, you can monitor the status of the input and output variables during program testing at the block "Program Press“ (→ )